SCIENCE CHINA
Information Sciences

• Supplementary File •

Foresighted Real-time Hierarchical Resource Scheduling in Dynamic Multi-domain Satellite Networks

Hongmei He[1], Di Zhou[1*], Min Sheng[1], Jiandong Li[1] & Chau Yuen[2]

[1]*State Key Laboratory of Integrated Service Networks, Xidian University, Xi'an* 710071*, China*
[2]*School of Electrical and Electronic Engineering, Nanyang Technological University* 639798*, Singapore*

Appendix A Related Work

Resource management plays a vital role in improving the scheduling capabilities of satellite systems for missions [3]. Several prominent efforts have mitigated resource underutilization and scheduling inefficiencies caused by the dynamics of ISLs and satellite-ground links (SGLs) inherent in satellite networks. These investigations fall into two main categories: non-real-time and real-time resource scheduling. Table A1 summarizes the research on satellite resource scheduling.

Non-real-time resource scheduling typically involves missions that can be accurately predicted in advance. In other words, the mission information is completely known. Various efforts have been made in designing non-real-time scheduling strategies for satellite networks. Concretely, an iterative optimization method was proposed in [4], which decomposed the satellite network resource allocation problem using the block coordinate descent method to enhance data transmission. Multi-dimensional resource collaboration problems for satellite networks can typically be modeled as ILP/MILP problems. Based on this, literature [5] addressed the intricate satellite network resource scheduling problem iteratively by proposing lightweight two-stage column generation algorithms, which optimize resource utilization while maximizing the successful completion rate of missions. In addition, literature [6] proposed an efficient inter-satellite cooperative computation offloading algorithm for low Earth orbit satellite networks, which optimizes resource allocation and offloading decisions to effectively reduce system latency and energy consumption. Furthermore, the recent work [7] formulated a joint data compression and task scheduling problem, which they efficiently solved using a semidefinite relaxation approach to enhance data offloading efficiency.

Generally, the aforementioned methods assume that the arrival information of the mission during the scheduling period can be accurately predicted. The resource allocation results for all time slots in the scheduling period are calculated in the network operation control center, after which the scheduling commands are sent to the satellite [5]. However, these non-real-time scheduling strategies cannot be directly applied to the real MDSN, which requires real-time scheduling responses for dynamic and unknown missions. Due to the random fluctuations of missions and resources in real satellite networks, real-time resource scheduling is practical and vital for improving the performance of satellite networks [9]. Some efforts have focused on designing algorithms for stochastic dynamic environments to overcome the limitations of non-real-time scheduling methods that require completely known mission information [1, 10–13].

Specifically, a mission-oriented real-time scheduling strategy for satellite networks was proposed in [10], which leveraged a random trial technique method to construct the revenue function and proposed an adaptive method to enhance the throughput and reduce costs. To address inefficient resource scheduling resulting from dynamic changes in mission priorities and satellite positions, an online autonomous satellite scheduling method was proposed in [11], which enhances the network revenue. In addition, [12] effectively integrated computational and communication resources, minimizing scheduling cost consumption through the application of graph theory and artificial intelligence. The most recent work [13] introduced a constrained multi-agent reinforcement learning dynamic routing algorithm, which effectively harmonized objective enhancement and constraint fulfillment during updates to policies and Lagrange multipliers. Furthermore, for resource scheduling in multi-layer networks, [1] proposed a multi-objective reinforcement learning-based routing strategy to meet diverse and differentiated service requirements.

Appendix B System Model and Problem Formulation

The primary objective of network resource scheduling is to maximize the number of completed missions in the MDSN through collaborative satellite operations. To support the formulation of the scheduling strategy, the system model is structured into three components: the mission model, channel model, and energy consumption model. Specifically, the mission model formalizes the representation of missions, the channel model captures the transmission capability of communication links, and the energy model quantifies power usage. Effective resource allocation ensures that missions and resources are appropriately matched, thereby improving mission completion performance.

* Corresponding author (email: zhoudi@xidian.edu.cn)

Table A1 Summary of research on satellite resource scheduling

Category	Reference	Main Approach	Dynamic Missions	Inter-Satellite Collaboration	Multi-Domain	Limitations
Non-Real-Time	[4], [5], [6], [7], [8]	Iterative optimization, Column generation, ILP/MILP	×	Partial	×	Cannot handle dynamic and unknown missions
Real-Time	[10], [11]	Random trial technique, Online autonomous scheduling	✓	✓	×	Lacks explicit handling of multi-domain scheduling
Real-Time	[12]	Graph theory and artificial intelligence	✓	Partial	×	Ignore inter-satellite collaboration and multi-domain characteristics
Real-Time	[13]	Constrained multi-agent reinforcement learning	✓	✓	×	Lacks modeling of multi-domain network architecture
Real-Time	[1]	Multi-objective reinforcement learning routing	✓	✓	×	Not focused on the multi-domain specifics of satellite networks

Table A2 The Key Notations

Notations	Description
$\mathcal{D}$, $\mathcal{N}_S$, $\mathcal{N}_G$, d_k	The MDSN; the set of satellites; the set of ground stations, the k-th domain.
$\mathcal{M}$, $\mathcal{M}_t^{d_k}$	The set of missions; the set of the arrived missions of domain d_k in time slot t.
$\mathcal{E}$, $\mathcal{E}_{ss}$, $\mathcal{E}_{sg}$, $e_{i^t j^t}$	The set of available communication links; the set of ISLs; the set of SGLs; the link from i to j in t-th time slot.
T, Δt, $\mathcal{T}$	The total number of time slots in the scheduling period; the length of time slot; scheduling period.
δ_s^m, δ_e^m, ϖ^m, κ^m	The origin of mission m; the destination of m; the amount of mission data; the amount of computational resources required for the mission.
E_{i^t}, Eng_{i^t}, Eng_i^{max}, η, $\mathcal{K}_i$	The energy consumption of satellite i at slot t; the battery state of satellite i at the beginning of slot t; the battery capacity; the depth of discharge; the upper limit of computing.
$w_{i^t j^t}^m$, $v_{i^t}^m$	The transmission variable from i to j in t-th time slot; the computing variable of satellite i in time slot t.
S^t, a^t, r^t, $r_t^{d_k}$	The state of satellite i in t-th time slot; the cross-domain action in t-th time slot; the scaled count of completed missions in domain d_k; the total reward in t-th time slot.
$H_{i^t j^t}$, $c_{i^t j^t}$	The data rate of link $e_{i^t j^t}$; the capacity of link $e_{i^t j^t}$.
$S_{\mathcal{B}}^t$,$S_{\mathcal{C}}^t$, $S_{\mathcal{D}}^t$, $S_{\mathcal{E}}^t$, $S_{\mathcal{F}}^t$	The storage state information; the communication state information; the energy state information; the mission number in each satellite; the amount of mission data in each satellite.

Appendix B.1 Mission Model

In general, a mission m can be represented as $m : [\delta_s^m, \delta_e^m, \varpi^m, \kappa^m]$, where δ_s^m and δ_e^m represent the origin and destination of m, respectively. Specifically, the mission origin δ_s^m indicates which domain the mission belongs to. The mission destination $\delta_e^m \in \mathcal{N}_G$ is the ground station. ϖ^m is the amount of mission data, which indicates the storage and communication resources required for the completion of the mission. κ^m denotes the amount of computational resources required for the mission. It is worth noting that each satellite is equipped with a computation unit, providing the computing capacity. Unlike the work [2], in line with the future trend of the integration of computing and networking [14], we allow computational resources to be shared among different domains. We assume that the satellite s_i is equipped with $\mathcal{K}_i$ CPU cores. Naturally, each satellite can serve at most $\mathcal{K}_i$ missions simultaneously [15].

The set of the arrived missions of domain d_k in time slot t is represented as $\mathcal{M}_t^{d_k}$, and all the missions in slot t constitute the set $\mathcal{M}_t$. Furthermore, we use $\mathcal{M}$ to represent the collection of all missions. For each mission m, we define that $w_{i^t j^t}^m$ is a binary variable indicating whether mission m is transmitted on link $e_{i^t j^t}$. If so, $w_{i^t j^t}^m = 1$; otherwise $w_{i^t j^t}^m = 0$. The variables $w_{i^t j^t}^m$ and $v_{i^t}^m$, where $i \in \mathcal{N}_S, j \in \mathcal{N}$, and $t \in \Gamma$ collectively define the realization path ϱ_m of mission m. Moreover, we assume that the mission is not detachable during scheduling. Therefore, the variable $w_{i^t j^t}^m$ should satisfy the constraint, that is,

$$\sum_{j \in \mathcal{N}} w_{i^t j^t}^m + w_{i^t i^{t+1}}^m \leqslant 1, \forall\, m \in \mathcal{M}, i \in \mathcal{N}_S, t \in \Gamma, i \neq j. \tag{B1}$$

Specifically, if the left side of the above constraint equals 1, it means that mission m is either stored on satellite i until the next time slot or sent from satellite i to j. If the left side of the above constraint equals 0, it means that mission m does not pass through these satellites. In addition, it is essential to satisfy the basic flow conservation constraints for each mission transmitted via satellites, ensuring a balanced inflow and outflow of mission data at every satellite. This guarantees the consistency of mission data and prevents any anomalous creation or loss of information, as expressed in (B2), where $w_{j^{t-1} j^t}^m = 1$ indicates that mission m is stored on satellite j

from the $(t-1)$-th to the t-th time slot.

$$\begin{cases} \sum\limits_{i\in\mathcal{N}_S} w^m_{i^t j^t} = \sum\limits_{i\in\mathcal{N}} w^m_{j^t i^t} + w^m_{j^t j^{t+1}} \ ,\forall\, m\in\mathcal{M}, j\in\mathcal{N}_S, t=1, i\neq j, \\ \sum\limits_{i\in\mathcal{N}_S} w^m_{i^t j^t} + w^m_{j^{t-1} j^t} = \sum\limits_{i\in\mathcal{N}} w^m_{j^t i^t} + w^m_{j^t j^{t+1}} \ ,\forall\, m\in\mathcal{M}, j\in\mathcal{N}_S, t\in\{2,3,...,T-1\}, i\neq j, \\ \sum\limits_{i\in\mathcal{N}_S} w^m_{i^t j^t} + w^m_{j^{t-1} j^t} = \sum\limits_{i\in\mathcal{N}} w^m_{j^t i^t} \ ,\forall\, m\in\mathcal{M}, j\in\mathcal{N}_S, t=T, i\neq j. \end{cases} \tag{B2}$$

Moreover, satellites are equipped with a buffer with limited capacity, which is responsible for storing data for subsequent forwarding, computing, etc. The total amount of data stored on satellite j cannot exceed its buffer capacity b_j^{max}, i.e.,

$$\sum_{m\in\mathcal{M}}\sum_{i\in\mathcal{N}_S} (w^m_{j^{t-1} j^t}\cdot\varpi^m + w^m_{i^t j^t}\cdot\varpi^m) \leqslant b_j^{max}, \forall\, j\in\mathcal{N}_S,\ t\in\Gamma. \tag{B3}$$

The completion of mission m requires not only transmitting the mission to its destination but also satisfying the computing resource requirements, i.e.,

$$\sum_{i\in\mathcal{N}_S}\sum_{t\in\Gamma} v^m_{i^t} = \sum_{i\in\mathcal{N}_S}\sum_{j\in\mathcal{N}_G}\sum_{t\in\Gamma} w^m_{i^t j^t}\cdot\kappa^m, \forall m\in\mathcal{M}, \tag{B4}$$

in which $v^m_{i^t}$ means whether mission m utilizes the computing resources on satellite i at t-th time slot. If so, $v^m_{i^t}=1$; otherwise, $v^m_{i^t}=0$. In addition, the consumption of computing resources to complete a mission must occur on the path of mission transmission. Therefore, we have the following constraint:

$$v^m_{i^t} \leqslant \sum_{j\in\mathcal{N}} w^m_{i^t j^t} + w^m_{i^t i^{t+1}}\ ,\ \ \forall\, i\in\mathcal{N}_S,\ t\in\Gamma,\ m\in\mathcal{M}. \tag{B5}$$

Furthermore, mission computing requirements on satellite i can not exceed the upper limit of computing capacity, i.e.,

$$\sum_{m\in\mathcal{M}} v^m_{i^t} \leqslant \mathcal{K}_i,\ \ \forall\, i\in\mathcal{N}_S,\ t\in\Gamma. \tag{B6}$$

Appendix B.2 Channel Model

Link rate is used to represent the transmission capability of the link and can be represented as [16]

$$H_{i^t j^t} = \begin{cases} \dfrac{P_{tr}\, G^i_{tr}\, G^j_{re}}{\upsilon\cdot\iota\cdot(E_b/N_0)_{\text{req}}\cdot L_m}\cdot\left(\dfrac{c}{4\pi\cdot\ell(e_{i^t j^t})\cdot f}\right), & \text{if } e_{i^t j^t}\in\mathcal{E}_{ss}, \\ 0, & \text{otherwise}, \end{cases} \tag{B7}$$

where P_{tr} is the satellite transmission power (in W), G^i_{tr} and G^j_{re} are the transmitting and receiving antenna gain of the link $e_{i^t j^t}$, respectively. In addition, υ and ι are the Boltzmann constant (in $\text{J}\cdot\text{K}^{-1}$) and the system total noise temperature (in K). $(E_b/N_0)_{req}$ is the required ratio of the energy received per bit to the noise density, and L_m is the link margin [17]. Furthermore, c is the speed of light (in km/s), f denotes the communication center frequency (in Hz) of the link $e_{i^t j^t}$, $\ell(e_{i^t j^t})$ represents the slant range (in km). The maximum amount of data that can be transmitted over the link $e_{i^t j^t}$ within a time slot is given by:

$$c_{i^t j^t} = H_{i^t j^t}\cdot\Delta t, \tag{B8}$$

in which Δt is the length of a time slot. Limited by the transmission capability of the link $e_{i^t j^t}$, the amount of mission data to be transmitted cannot exceed the link capacity, i.e.,

$$\sum_{m\in\mathcal{M}} w^m_{i^t j^t}\cdot\varpi^m \leqslant c_{i^t j^t}, \forall\, i\in\mathcal{N}_S, j\in\mathcal{N}, t\in\Gamma. \tag{B9}$$

Appendix B.3 Energy Consumption Model

The satellite energy consumption model includes both static and dynamic energy components. Static energy consumption accounts for the energy required to sustain the operation of the basic satellite system and can be denoted as $E^O_{i^t} = P^o\cdot\Delta t$, where P^o represents the static power of the satellite. Dynamic energy consumption represents the energy consumption caused by satellites transmitting,

receiving, and computing mission data. The energy consumed by the satellite mission data transmitting $E_{i^t}^{TR}$, which includes the energy consumed by the satellite to transmit data through SGLs and ISLs, can be expressed as

$$E_{i^t}^{TR} = \sum_{m\in M}\left[\sum_{j\in\mathcal{N}_S, e_{i^tj^t}\in\mathcal{E}_{ss}} P_{ss}^{T}\cdot\frac{w_{i^tj^t}^{m}\cdot\varpi^m}{c_{i^tj^t}}\cdot\Delta t + \sum_{j\in\mathcal{N}_G, e_{i^tj^t}\in\mathcal{E}_{sg}} P_{sg}^{T}\cdot\frac{w_{i^tj^t}^{m}\cdot\varpi^m}{c_{i^tj^t}}\cdot\Delta t\right], \forall i\in\mathcal{N}_S, t\in\Gamma, \tag{B10}$$

where P_{ss}^{T} and P_{ss}^{R} are the transmission power of the ISL and the SGL, respectively. The energy required by the satellite to receive mission data is $E_{i^t}^{RE}$, which can be represented as

$$E_{i^t}^{RE} = \sum_{m\in\mathcal{M}}\left[\sum_{j\in\mathcal{N}_S, e_{i^tj^t}\in\mathcal{E}_{ss}} P_{ss}^{R}\cdot\frac{w_{i^tj^t}^{m}\cdot\varpi^m}{c_{i^tj^t}}\cdot\Delta t\right], \forall i\in\mathcal{N}_S, t\in\Gamma, \tag{B11}$$

where P_{ss}^{R} is the power of the satellite to receive data via ISL. In addition, the cross-domain transmission of mission data requires additional energy consumption, which is

$$E_{i^t}^{C} = \sum_{m\in\mathcal{M}}\sum_{j\in\mathcal{N}_S|i, e_{i^tj^t}\in\mathcal{E}_{ss}} \omega_c\cdot w_{i^tj^t}^{m}\cdot\varpi^m, \forall i\in\mathcal{N}_S, t\in\Gamma, \tag{B12}$$

where ω_c denotes the energy consumption required per unit of cross-domain mission data, the $\mathcal{N}_S|i$ denotes the set of satellites in $\mathcal{N}_S$ excluding those that belong to the same domain as satellite i. Therefore, the total energy consumption of satellite i at slot t is expressed as

$$E_{i^t} = E_{i^t}^{O} + E_{i^t}^{TR} + E_{i^t}^{RE} + E_{i^t}^{C}. \tag{B13}$$

In addition, the remaining energy of the satellite cannot be greater than its battery energy protection threshold, i.e.,

$$Eng_{i^t} - E_{i^t} \geqslant Eng_i^{\max}(1-\eta), \forall i\in\mathcal{N}_S, t\in Gamma, \tag{B14}$$

where Eng_{i^t} is the battery state of satellite i at the beginning of slot t, and $\eta\in[0,1]$ is the depth of discharge.

Appendix B.4 Problem Formulation

We construct the problem $\mathcal{P}0$ to maximize the number of scheduled missions in MDSN, under the relevant constraints.

$$\begin{aligned}
\mathcal{P}0: \max & \sum_{t\in\Gamma}\sum_{m\in\mathcal{M}_t}\sum_{i\in\mathcal{N}_S}\sum_{j\in\mathcal{N}_G} w_{i^tj^t}^{m} \\
\text{s.t.} & \text{flow conservation constraints (B1), (B2)}, \\
& \text{storage and link capacity constraints (B3), (B9)}, \\
& \text{energy \& computing constraints (B4), (B5), (B6), (B14)}, \\
& w_{i^tj^t}^{m}\in\{0,1\}, \forall\, m\in\mathcal{M}, i\in\mathcal{N}_S, j\in\mathcal{N}, t\in\Gamma, \qquad \text{(B15)} \\
& v_{i^t}^{m}\in\{0,1\}, \forall\, m\in\mathcal{M}, i\in\mathcal{N}_S, t\in\Gamma. \qquad \text{(B16)}
\end{aligned}$$

Taking into account the timeliness requirement of the mission, it is imperative to complete it within the designated time frame. Consequently, the scheduling process is structured into a series of successive rolling time windows, each with a length of T_w. As a result, the problem $\mathcal{P}1$ can be derived as

$$\begin{aligned}
\mathcal{P}1: \max & \sum_{t\in\Gamma}\sum_{m\in\mathcal{M}_t}\sum_{\tau\in T_w^t}\sum_{i\in\mathcal{N}_S}\sum_{j\in\mathcal{N}_G} w_{i^\tau j^\tau}^{m}, \\
\text{s.t.} & (\text{B1})-(\text{B6}), (\text{B9}), (\text{B14}), \forall i\in\mathcal{N}_S, j\in\mathcal{N}, \tau\in T_w^t, \\
& w_{i^\tau j^\tau}^{m}\in\{0,1\}, \forall m\in\mathcal{M}, i\in\mathcal{N}_S, j\in\mathcal{N}, \tau\in T_w^t, \qquad \text{(B17)} \\
& v_{i^\tau}^{m}\in\{0,1\}, \forall m\in\mathcal{M}, i\in\mathcal{N}_S, \tau\in T_w^t, \qquad \text{(B18)}
\end{aligned}$$

in which T_w^t represents the slot from t-th to $(t+T_w)$-th time slot.

It is worth noting that in highly dynamic MDSN, accurately predicting the set of missions throughout the scheduling period T is impractical due to the unpredictability of the missions. Consequently, solving the problem $\mathcal{P}1$ directly using linear programming methods is unfeasible. Given that the unpredictable arrival missions dynamically impact the allocation of buffers, satellite power, and opportunities for inter-satellite communication, effective resource scheduling becomes pivotal in enhancing MDSN mission completion. Thus, our goal is to adaptively schedule resources in response to environmental dynamics and unpredictable mission arrivals, thereby maximizing mission completion in the MDSN over the scheduling period.

Appendix C Framework and Algorithm Design

To clarify how the proposed FRHRS framework functions, this section first introduces the overall hierarchical architecture and then presents the specific algorithms used in each layer. The framework comprises an upper layer that makes strategic cross-domain scheduling decisions and a lower layer that performs mission realization path planning accordingly. At each time slot, the upper layer evaluates whether global cross-domain scheduling should be triggered, while the lower layer constructs a feasible realization path, allocates resources along it, and returns execution feedback. Through this hierarchical interaction, the upper layer focuses on long-term scheduling strategy, whereas the lower layer handles fine-grained operational decisions. The following subsections describe these components in detail.

Appendix C.1 FRHRS Framework

In this section, we propose a novel FRHRS framework consisting of an upper layer and a lower layer. Rather than consistently performing global cross-domain resource scheduling, as is done in most existing research on MDSNs, we design a learning-based dynamic scheduling algorithm in the upper layer to make real-time decisions on whether to trigger cross-domain scheduling at each time slot. On one hand, non-cross-domain scheduling can lead to the wastage of fragmented resources within the domain, resulting in the risk of a low rate of mission completion. On the other hand, although cross-domain scheduling can fully utilize multi-domain resources, it incurs additional cross-domain data computing and energy consumption, impacting future resource scheduling. Therefore, our goal is to achieve long-term optimization of resource scheduling by balancing cross-domain and non-cross-domain scheduling.

In the lower layer, we focus on matching the dynamic multidimensional resource with the current mission, i.e., the problem of resource allocation for mission realization path (MRP-RA), which is the static subproblem of the dynamic scheduling problem in the upper layer. The MRP-RA problem, aimed at optimizing the number of completed missions, requires detailed realization path planning for each mission. The MRP-RA problem can be explicitly represented as an ILP problem, but is time-consuming to solve directly due to the large number of variables and complex practical constraints. Considering that scheduling response lags in dynamic networks can lead to scheduling failures, we design a rapid scheduling algorithm that maximizes the number of completed missions by minimizing the implementation cost of each mission. Subsequently, the results of the MRP-RA problem are quickly fed back to the upper layer. Finally, the number of completed missions for the whole system is maximized by dynamic mission scheduling with iterations of upper and lower layers. In the following subsections, we will detail the designed resource scheduling methods in the upper and lower layers, respectively.

Appendix C.2 Real-time Intelligent Cross-domain Decision Algorithm in the Upper Layer

Due to the differing ranges of resources available for cross-domain scheduling and non-cross-domain scheduling, and the additional overhead introduced by cross-domain scheduling, we further refine the resource allocation in $\mathcal{P}1$, resulting in the formulation of problem $\mathcal{P}2$.

$$\mathcal{P}2:\ \max \sum_{t\in\Gamma} \begin{cases} \mathcal{P}_t^{NCD}: & \sum\limits_{d_k\in\mathcal{D}} \sum\limits_{m\in\mathcal{M}_t^{d_k}} \sum\limits_{\tau\in T_w^t} \sum\limits_{i\in\mathcal{N}_S^{d_k}} \sum\limits_{j\in\mathcal{N}_G} w_{i^\tau j^\tau}^m, \\ \mathcal{P}_t^{CD}: & \sum\limits_{m\in\mathcal{M}_t} \sum\limits_{\tau\in T_w^t} \sum\limits_{i\in\mathcal{N}_S} \sum\limits_{j\in\mathcal{N}_G} w_{i^\tau j^\tau}^m, \end{cases}$$
$$\text{s.t.}\quad (\text{B1})-(\text{B6}),(\text{B9}),(\text{B14}),(\text{B17}),(\text{B18}),\forall m\in\mathcal{M}, i\in\mathcal{N}_S, j\in\mathcal{N}, \tau\in T_w^t.$$

It's important to note that problem $\mathcal{P}2$ has tighter bounds than $\mathcal{P}1$ because of the imposition of more stringent constraints of missions and multidimensional resources. At each time slot, we obtain the subproblem P_t^{NCD} or P_t^{CD} based on the cross-domain scheduling decision made in the upper layer. Therefore, the overall dynamic MDSN scheduling problem $\mathcal{P}1$ can be divided into a series of static subproblems. However, it is still intractable to solve it directly due to the dynamic arrival of missions. In practice, determining whether to implement cross-domain scheduling to optimize mission completion throughout the scheduling period is a sequential decision problem that boils down to a Markov Decision Process (MDP). We designate the beginning of each time slot as a decision point. Depending on whether each decision point performs cross-domain or intra-domain scheduling, the whole MDSN can be dynamically divided into a series of static subproblems, each of which contains different scopes of missions and resources within the domain and across the domain. For the upper layer, missions are scheduled intra-domain until the upper layer agent decides to schedule across the domain at time slot t.

State: In the t-th time slot, the state of MDSN can be defined as

$$S^t \triangleq \{S_{\mathcal{B}}^t, S_{\mathcal{C}}^t, S_{\mathcal{D}}^t, S_{\mathcal{E}}^t, S_{\mathcal{F}}^t\}, \tag{C1}$$

which consists of the storage state information $S_{\mathcal{B}}^t$, communication state information $S_{\mathcal{C}}^t$, energy state information $S_{\mathcal{D}}^t$, and mission state information $S_{\mathcal{E}}^t$, $S_{\mathcal{F}}^t$ of all satellites in the MDSN. The storage state $S_{\mathcal{B}}^t$ of the MDSN is composed of the memory states $s_{\mathcal{B}}^{t,i}$ of each satellite. The continuous variable $s_{\mathcal{B}}^{t,i}\in[0, CB_{\max}^i]$ denotes the remaining available memory of satellite i at the t-th time slot; $S_{\mathcal{C}}^t$ is composed of the communication state of each satellite in the MDSN, i.e., $S_{\mathcal{C}}^t:\{s_{\mathcal{C}}^{t,i}|i\in\mathcal{N}_S\}$, where $s_{\mathcal{C}}^{t,i}$ is the current sum of each satellite's remaining available link capacity. $S_{\mathcal{D}}^t$ represents the energy state, which is composed of the energy state of each satellite, i.e., $S_{\mathcal{D}}^t:\{s_{\mathcal{D}}^{t,i}|i\in\mathcal{N}_S\}$, where $s_{\mathcal{D}}^{t,i}\in[0, EB_{\max}^i]$ is the current remained on-board battery power in the satellite; $S_{\mathcal{E}}^t$ and $S_{\mathcal{F}}^t$ is the mission state, among which $S_{\mathcal{E}}^t:\{s_{\mathcal{E}}^{t,i}|i\in\mathcal{N}_S\}$ denotes the mission number in each satellite and $S_{\mathcal{F}}^t:\{s_{\mathcal{F}}^{t,i}|i\in\mathcal{N}_S\}$ denotes the amount of data for each mission in each satellite. The state transition function is denoted as $\mathcal{TR}:\mathcal{S}\times\mathcal{A}\times\mathcal{S}\to[0,1]$, which represents the probability of the system evolving from one state to another given the actions taken by the agents. Given the complexity of the system dynamics and the difficulty in precisely modeling the transition probabilities, this chapter employs a model-free reinforcement learning approach. In this approach, neural networks are utilized to train and predict the cumulative future rewards.

Action: Our objective is to maximize the number of completed missions throughout the entire scheduling period of MDSN. To achieve this, we establish a rule that the immediate reward for an action in terms of cross-domain scheduling taken at the t-th slot corresponds to the number of completed missions resulting from that specific scheduling action. Following this rule, the sum of the immediate rewards constitutes the total objective of the entire scheduling problem. Such reward shaping encourages the upper layer to optimize the overall dynamic mission scheduling problem from a long-term perspective. The cross-domain action in t-th time slot is represented as $a_t \in \mathcal{A}$, where the binary action space $\mathcal{A} \in \{0, 1\}$, corresponds to the approaches of two types of subproblems, i.e., intra-domain scheduling and cross-domain scheduling.

Reward: The reward is obtained by solving the subproblems generated by the actions in the upper layer. Specifically, if $a_t = 0$, the reward is achieved by intra-domain scheduling and can be denoted as $r_t = \sum_{d_k \in \mathcal{D}} r_t^{d_k}$, where $r_t^{d_k}$ denotes the scaled count of completed missions in domain d_k. $r_t^{d_k}$ is obtained by solving the following:

$$\begin{aligned}\mathcal{P}_{t,d_k}^{NCD}: \max & \sum_{m\in\mathcal{M}_t^{d_k}} \sum_{\tau\in T_w^t} \sum_{i\in\mathcal{N}_S^{d_k}} \sum_{j\in\mathcal{N}_G} w_{i^\tau j^\tau}^m \\ \text{s.t. } & \text{flow conservation constraints}: (\text{B1}), (\text{B2}), \\ & \forall\, i \in \mathcal{N}_S^{d_k}, j \in \mathcal{N}_G \cup \mathcal{N}_S^{d_k}, \tau \in T_w^t, \\ & \text{storage \& link capacity constraints}: (\text{B3}), (\text{B9}), \\ & \forall\, i \in \mathcal{N}_S^{d_k}, j \in \mathcal{N}_G \cup \mathcal{N}_S^{d_k}, \tau \in T_w^t, \\ & \text{energy \& computing constraints}: (\text{B4}), (\text{B5}), \\ & (\text{B6}), (\text{B14}), \forall\, i \in \mathcal{N}_S^{d_k}, j \in \mathcal{N}_G \cup \mathcal{N}_S^{d_k}, \tau \in T_w^t, \\ & w_{i^\tau j^\tau}^m \in \{0,1\}, \forall m \in \mathcal{M}_t^{d_k}, i \in \mathcal{N}_S^{d_k}, j \in \mathcal{N}_G \cup \mathcal{N}_S^{d_k}, \tau \in T_w^t, \\ & v_{i^\tau}^m \in \{0,1\}, \forall m \in \mathcal{M}_t^{d_k}, i \in \mathcal{N}_S^{d_k}, \tau \in T_w^t. \end{aligned} \tag{C2}$$

In addition, if $a_t = 1$, r_t is the reward for cross-domain scheduling, which is solved by the following:

$$\begin{aligned}\mathcal{P}_t^{CD}: \max & \sum_{m\in\mathcal{M}_t} \sum_{\tau\in T_w^t} \sum_{i\in\mathcal{N}_S} \sum_{j\in\mathcal{N}_G} w_{i^\tau j^\tau}^m \\ \text{s.t. } & (\text{B1}) - (\text{B6}), (\text{B9}), (\text{B14}), (\text{B17}), (\text{B18}), \forall m \in \mathcal{M}_t, i \in \mathcal{N}_S, j \in \mathcal{N}, \tau \in T_w^t, \end{aligned} \tag{C3}$$

We propose the RICD algorithm to solve the dynamic RT-CDRS in the upper layer of MDSN. Specifically, we formulate the sequential decision-making of resource scheduling as an MDP, where changes in satellite resources and missions can be viewed as state transfers in the MDP. We efficiently update the weight matrix to learn the optimal cross-domain resource scheduling policy. The MDSN decides its current scheduling action, i.e., whether to implement cross-domain collaborative scheduling, based on the current multidimensional resource state and mission state to maximize the number of completed missions and offload to $\mathcal{N}_G$ across the MDSN.

The mapping from the MDSN state to action in the t-th time slot can be denoted as $\theta_t : s_t \to a_t$. The continuous mapping over a period is referred to as a policy, which can be represented as

$$\xi = \{\theta_1, \theta_2, \cdots, \theta_T\}. \tag{C4}$$

The policy dictates whether the MDSN performs cross-domain scheduling or intra-domain scheduling for the current mission based on the observed states at the current time slot. The cumulative expected reward of MDSN within T time slots under the state s_t and a policy ξ can be expressed as

$$\mathcal{R}(s_t, \xi) = \mathbb{E}\left[\sum_{m=0}^{T-t} \gamma^m \cdot r(s_{t+m}, \theta_{t+m}^{\xi}(s_{t+m}))\right], \tag{C5}$$

where, $\mathbb{E}[\cdot]$ denotes the expectation function, γ represents the discount factor. The goal of resource scheduling is to find the optimal policy ξ^* to maximize the number of completed missions in MDSN, that is,

$$\xi^* = \underset{\xi\in\Xi, \mathbf{s}, \mathbf{a}, \mathbf{r}}{\arg\max}\, \mathcal{R}(s_0, \xi), \tag{C6}$$

where Ξ is the set of all feasible strategies.

Among various DRL algorithms, we employ the dueling double deep Q-learning network (D3QN) due to its superior performance and the straightforward structure [18]. The D3QN algorithm integrates the concepts of double DQN [19] and dueling DQN [20], both of which are extensions of the Deep Q-Learning Network (DQN) [21]. Our upper-layer solution does not aim to schedule the exact best path for mission realization from dynamic multidimensional resources. Rather, we find trade-offs between dynamic cross-domain and non-cross-domain using the D3QN model. D3QN can decouple the choice of actions from the objective value function compared to DQN, enabling the network to learn the environment better [22]. By separating the value of states from the advantage of actions, the

Algorithm C1 RICD algorithm for cross-domain decision.

Input: Topology $\mathcal{E}$ of MDSN obtained through satellite simulation, scheduling period T during an episode
Output: Trained upper layer resource scheduling policy ξ_ϑ for MDSN

Initialization: the parameters of the online Q-network ϑ_c, set the target Q-network with $\vartheta_t \leftarrow \vartheta_c$, the replay buffer D
for *Episode* $= 1, 2, \cdots, \mathcal{L}$ **do**
  Obtain the initial state s of the MDSN
  for *Time slot* $t = 0, 1, \cdots, T$ **do**
    Select the scheduling action a_t using the ϵ-greedy strategy based on the current online Q-network $\arg\max_a Q(s, a, \vartheta_c)$.
    if $a_t = 0$ **then**
      Get the non-cross domain reward r^t by solving the problem (C2)
    else
      Get the cross-domain reward r^t by solving the problem (C3)
    end if
    Obtain realization path ϱ_m for each mission by Algorithm C2
    Update the state s_{t+1} of the MDSN by ϱ_m
    Store the cross-domain decision trajectory $\zeta_t = \{s_t, a_t, r_t, s_{t+1}\}$ in D
    Sample a mini-batch of (s_j, a_j, r_j, s_{j+1}) from the experience buffer D
    for each sampled tuple (s_j, a_j, r_j, s_{j+1}) **do**
      Compute the loss according to equation (C8)
    end for
    Accumulate the losses and update ϑ_c via gradient descent
    Every fixed number of steps, copy the online Q-network parameters ϑ_c to the target Q-network ϑ_t
  end for
end for

network can learn combinations of values and advantages more flexibly, thereby improving performance in complex environments. The Q-values, including the value function Q_v and the advantage function Q_a, can be expressed as follows:

$$Q(s,a) = Q_v(s) + [Q_a(s,a) - \frac{1}{|\mathcal{A}|}\sum\nolimits_{a' \in \mathcal{A}} Q_a(s,a')], \tag{C7}$$

in which, $|\mathcal{A}|$ represents the size of the action space.

To further improve training stability and mitigate the instability caused by rapidly changing Q-values, D3QN adopts a dual-network architecture. Specifically, an online Q-network Q^{ϑ_c} (with parameters ϑ_c) is used to evaluate the Q-values under the current policy, while a separate target network Q^{ϑ_t} is employed to compute stable target Q-values and loss functions. The parameters of the target network are periodically synchronized with those of the online network to ensure training stability.

The online network selects the scheduling action according to $\arg\max_a Q(s, a, \vartheta^c)$, while the target network evaluates the expected effect of the selected action in subsequent resource scheduling, i.e., $Q(s, a, \vartheta_t)$, where $a = \arg\max_a Q(s, a, \vartheta^c)$. The target Q-value is defined as: $y = r + \gamma Q(s, \arg\max_a Q(s, a, \vartheta_c), \vartheta_t)$ Accordingly, the loss function is formulated as:

$$Loss(\vartheta_c) = \mathbb{E}[(y - Q(s, a, \vartheta_c))^2]. \tag{C8}$$

The parameters ϑ_c are then updated by minimizing the loss using gradient descent. To ensure the stable evolution of the resource scheduling policy, the parameters of the online network are periodically copied to the target network, i.e., $\vartheta_t \leftarrow \vartheta_c$.

The pseudocode of the RICD algorithm is presented in Algorithm C1. In each training iteration, the agent begins from the initial state s_0 and selects a scheduling action a_t for each time slot t according to the current policy ξ_ϑ. If $a_t = 0$, the agent obtains the immediate reward for intra-domain scheduling by solving problem (C2); if $a_t = 1$, the reward for cross-domain scheduling is obtained by solving problem (C3). Subsequently, Algorithm C2 determines the actual realization path ϱ_m for each mission, based on which the system state is updated to s_{t+1}. Meanwhile, each experience tuple s_t, a_t, r_t, s_{t+1} is stored in the experience replay buffer D. Training samples are then randomly drawn from D, and the loss is minimized using gradient descent to iteratively update the policy parameters ϑ_c. This process continues until convergence to the optimal cross-domain scheduling policy ξ_ϑ, which is represented as $\xi_\vartheta(s) = \arg\max_a Q(s, a; \vartheta_c)$, which selects the action with the highest Q-value under the given state s.

Appendix C.3 Mission Realization Path Planning in the Lower Layer

In the lower layer, we focus on rapidly identifying a feasible realization path for a specific mission, based on the cross-domain decision made in the upper layer. The realization path of a mission includes a sequence of inter-satellite transmissions and the selection of computing nodes, represented by the variables $v_{i^t}^m$ and $w_{i^t j^t}^m$, which jointly define how mission m is progressively processed and

Algorithm C2 RMC2P algorithm for mission path planning.

Input: The link capacity $c_{i^t j^t}$, the satellite computing capacity $\mathcal{K}_i$, the length of T_w, the arrived missions in the current time slot, and the set of completed missions $\Omega = \varnothing$.

Output: Completed missions and the availability of multidimensional resources.

1: Sort missions in descending order according to the amount of mission data.
2: **for** Each mission in $\mathcal{M}_t/\mathcal{M}_t^{d_k}$ (cross-domain / intra-domain) **do**
3: Construct a minimum-cost implementation problem for mission m that satisfies the multidimensional resource constraints with T_w^t, $\mathcal{K}_i$, and $c_{i^t j^t}$.
4: Solve the problem $\mathcal{P}3$ obtain the realization path ϱ_m of mission m.
5: **if** $\varrho_m \neq \emptyset$ **then**
6: Add the completed mission m to set Ω.
7: Update network resources based on the realization path ϱ_m.
8: **end if**
9: **end for**
10: Obtain the number of missions completed in the current time slot.

forwarded across the satellite network. In the lower layer scheduling process, it is required to 1) determine the mission realization path, 2) calculate the remaining resource state, and 3) provide the upper layer with the reward for a decision taken according to the current states. The lower layer scheduling problem (i.e., the MRP-RA problem) consists of two ILP problems: $\mathcal{P}_t^{NCD}$ and $\mathcal{P}_t^{CD}$. These problems are slow to solve due to numerous variable constraints, leading to difficulty in responding quickly to MRP-RA. Therefore, we propose a rapid mission realization path planning (RMC2P) algorithm with low complexity to address the problem in the lower layer.

To ensure efficient transmission of mission data to ground stations, it is essential to optimize the mission scheduling strategy under resource constraints. To this end, we construct an objective function $\sigma(m)$ for each mission m, which comprehensively accounts for the cost of link, storage, and computing, and use it as the optimization objective:

$$\begin{aligned}\mathcal{P}3: \min &\sigma(m) = \min\left[\varsigma_B^m + \varsigma_T^m + \varsigma_C^m\right] \\ &\text{s.t.}(\text{B1}) - (\text{B6}), (\text{B9}), (\text{B14}), (\text{B15}), (\text{B16}), \forall m \in \mathcal{M}_t,\end{aligned}$$

where ς_B^m is the storage cost, which can be further expressed as:

$$\varsigma_B^m = \begin{cases} \beta_B(\sum\limits_{i\in\mathcal{N}_S^{d_k}} \sum\limits_{\tau\in T_w^t} w_{i^\tau i^{\tau+1}}^m), \forall m \in \mathcal{M}_t,\ d_k \in \mathcal{D}, & \text{if } a_t = 0, \\ \beta_B(\sum\limits_{i\in\mathcal{N}_S} \sum\limits_{\tau\in T_w^t} w_{i^\tau i^{\tau+1}}^m), \forall m \in \mathcal{M}_t, & \text{otherwise}, \end{cases} \tag{C9}$$

where β_B is the weight coefficient of storage cost. Similarly, ς_T^m and ς_C^m represent the link cost and computing cost, respectively,

$$\varsigma_T^m = \begin{cases} \beta_T(\sum\limits_{i\in\mathcal{N}_S^{d_k}} \sum\limits_{\substack{j\in\mathcal{N}\\ i\neq j}} \sum\limits_{\tau\in T_w^t} w_{i^\tau j^\tau}^m), \forall m \in \mathcal{M}_t,\ d_k \in \mathcal{D}, & \text{if } a_t = 0, \\ \beta_T(\sum\limits_{i\in\mathcal{N}_S} \sum\limits_{\substack{j\in\mathcal{N}\\ i\neq j}} \sum\limits_{\tau\in T_w^t} w_{i^\tau j^\tau}^m), \forall m \in \mathcal{M}_t, & \text{otherwise}, \end{cases} \tag{C10}$$

$$\varsigma_C^m = \begin{cases} \beta_C(\sum\limits_{i\in\mathcal{N}_S^{d_k}} v_{i^\tau}^m), \forall m \in \mathcal{M}_t,\ d_k \in \mathcal{D}, & \text{if } a_t = 0, \\ \beta_C(\sum\limits_{i\in\mathcal{N}_S} v_{i^\tau}^m), \forall m \in \mathcal{M}_t, & \text{otherwise}, \end{cases} \tag{C11}$$

in which the β_T and β_C are the weight coefficients of link and computing cost, respectively.

We aim to enhance mission completion by reducing the cost of storage, link, and computing for each mission. The pseudocode of the RMC2P algorithm is presented in Algorithm C2. This algorithm aims to improve the mission completion rate by minimizing the resource cost for each mission. Meanwhile, the realization of each mission must satisfy the basic flow constraints, resource constraints, and binary constraints. Problem $\mathcal{P}3$ is solved sequentially for each mission m to determine its realization path ϱ_m, which consists of the variables $w_{i^t j^t}^m$ and $v_{i^t}^m$. Given the realization path ϱ_m, the associated resource consumption can be obtained, and the availability of multidimensional resources is subsequently updated.

Within this framework, the lower-layer optimization problem is treated as part of the environment dynamics. The objective of the upper-layer agent is to learn to select actions a_t that lead to high rewards in state s_t, even though these rewards must be obtained indirectly through lower-layer optimization. It is important to note that the reward function is derived from the optimization outcomes of the lower-layer system. Therefore, it represents not only the direct consequence of the upper-layer action a_t but also embodies the influence of the lower-layer system state s_t and the performance of its optimizer. This design intentionally enables the upper-layer agent to learn macroscopic decision-making capabilities in complex environments.

Table D1 Main Parameters of Mission in MDSN

Domain	d_1	d_2	d_3	d_4	d_5	d_6
Mission computing demand	10 units	10 units	40 units	30 units	10 units	30 units
Mission data volume (Gbit)	[2∼6]	[2∼8]	[1∼2]	[0.2∼1]	[2∼6]	[0.3∼1]
Mission service type	Remote sensing	Remote sensing	IoT	IoT	Remote sensing	IoT

Although this work adopts a centralized single-agent RL framework with access to global system states, we acknowledge that collecting detailed real-time information (e.g., energy, storage, computation load, and mission status of all satellites) may introduce non-negligible signaling overhead in a practical network. Distributed or multi-agent approaches can alleviate such communication burdens and improve scalability by relying primarily on local information. However, they also face challenges such as partial observability, coordination among agents, and training instability. Our centralized design serves as a performance upper bound and a clean baseline for analyzing inter-satellite resource coupling. Extending the system toward a communication-efficient distributed RL architecture represents an important direction for future research.

Appendix C.4 Complexity Analysis

The complexity of the upper-level primarily stems from the forward propagation of the neural network during decision-making. For a network with L_{in} input neurons, one hidden layer containing L_{hidden} neurons, and L_{out} output neurons, the number of floating-point operations in a single forward propagation is mainly determined by matrix multiplications: $\mathcal{O}_{\text{upper}} = \mathcal{O}(L_{in} \cdot L_{hidden} + L_{hidden} \cdot L_{out})$. In the lower-level, when solving the specific implementation path for each mission, the corresponding problem $\mathcal{P}3$ is formulated as a small-scale ILP problem. In the worst-case scenario, its time complexity is $\mathcal{O}_{\text{lower}}(V' \cdot L' \cdot 2^{V'} \cdot |\mathcal{M}_t|)$, where $V' = T_w \cdot |\mathcal{N}_S| \cdot (|\mathcal{N}| + 1)$ represents the number of variables in $\mathcal{P}3$, and $L' = T_w \cdot |\mathcal{N}_S| \cdot (2|\mathcal{N}| + 7) + 1$ denotes the number of constraints. Overall, the complexity is mainly influenced by the lower-level scheduling. Although the complexity $\mathcal{O}(V' \cdot L' \cdot 2^{V'} \cdot |\mathcal{M}_t|)$ grows exponentially with respect to the number of variables V', both V' and the total term $V' \cdot L'$ are significantly smaller compared to those in $\mathcal{P}2$, whose complexity is $\mathcal{O}(V \cdot L \cdot 2^V)$, in which $V = T_w \cdot |\mathcal{N}_S| \cdot (|\mathcal{N}| + 1) \cdot |\mathcal{M}_t|$, $L = T_w \cdot |\mathcal{N}_S| \cdot (3|\mathcal{M}_t| + 3 + (2|\mathcal{M}_t| + 1)|\mathcal{N}|)$.

Appendix D Simulation Results and Discussion

In this section, we perform a comprehensive series of simulations using satellite parameters derived from the MDSN satellite simulator. Initially, we describe the simulation scenarios and algorithmic parameters. Subsequently, we validate the efficacy of the proposed foresighted resource scheduling method across various performance metrics and network configurations.

Appendix D.1 Simulation Configuration

We conduct simulations in the MDSN with up to six different domains, which contain 24, 12, 12, 16, 20, and 24 satellites, respectively. Specifically, satellites in d_1 are distributed at a height of 780km and an inclination of 45°. In d_2, satellites are distributed at a height of 1275km and an inclination of 53°. Similarly, d_3 operates at an altitude of 550 km, also with an inclination of 53°. For d_4, satellites are deployed at 1400 km with an inclination of 48°, while d_5 features satellites at 1414 km and 52°. d_6 includes satellites at 610 km with an inclination of 42°. Seven GSs are located at Beijing (40°N,116°E), Florida Ground Station (29°N,81°W), Kashi (39.5°N,76°E), Sanya (18°N,109.5°E), Frankfurt (50.12°N,9.92°E), Punta Arenas(53°S,71°W), Perth(31.8°S,115.89°E), respectively. We built up such an MDSN with the above parameters based on Matlab and a satellite simulator. The simulation scheduling horizon ranges from 12 Feb. 2024 04:00:00 to 13 Feb. 2024 04:00:00. The satellite orbital parameters adopt standard operational configurations, and transmission and receive powers follow commonly used settings.The missions are categorized into two main types: remote sensing and IoT [24, 25]. Within each main category, the missions are further subdivided into smaller categories based on their specific computing and data volume requirements, serving as missions for the services in each domain. Table D1 summarizes the mission characteristics in different domains. In this work, each domain is characterized by its dominant and most discriminative mission feature (e.g., large data volume for remote sensing versus computational intensity for IoT). This abstraction highlights the core inter-domain differences.

A 200-second slot is applied to approximate a stable topology while retaining key dynamics. The learning rate of 10^{-3} ensures stable RL training, and a discount factor of 0.99 highlights long-term scheduling performance. The online Q-network is implemented as a two-layer ReLU-based MLP with 256 neurons in the hidden layer. The establishment of the MDSN environment and the simulation of the framework and algorithm are operated in Python 3.8. We utilize Pytorch version 1.7.1 to build and train the neural networks. The optimization tools GUROBI 9.1.1 are also employed in the simulation to solve the *step* 4 in Algorithm C2.

Appendix D.2 Simulation Benchmark Design

Furthermore, to verify the performance of the proposed RICD-RMC2P algorithm in the MDSN, we present simulation results by comparing our work with the following benchmark algorithms.

• **Domain isolation column generation resource scheduling (DICG)** [23], [5]. In the MDSN, domains are isolated from each other and do not collaborate at the upper layer. At the lower layer, we adopt the state-of-the-art satellite resource scheduling algorithm [5], which is based on column generation: it decomposes the lower-layer scheduling problem into a master problem and multiple subproblems, and solves them iteratively.

• **Myopic-cross-domain column generation resource scheduling (MCDCG)** [2]. MCDCG makes a myopic decision on whether to schedule across domains in each time slot. At every communication opportunity, it greedily performs cross-domain scheduling to complete as many missions as possible using the currently available resources. At the lower layer, it employs the same state-of-the-art satellite resource scheduling algorithm as in DICG.

• **Optimal satellite resource allocation (OSRA).** OSRA solves the lower layer scheduling problem directly via the advanced solver Gurobi.

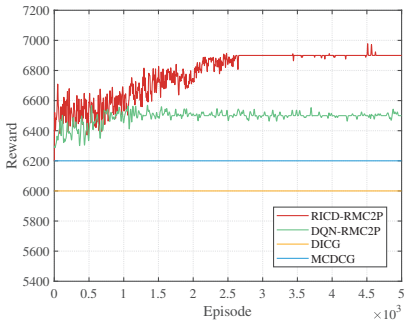

Figure D1 The reward in the training process.

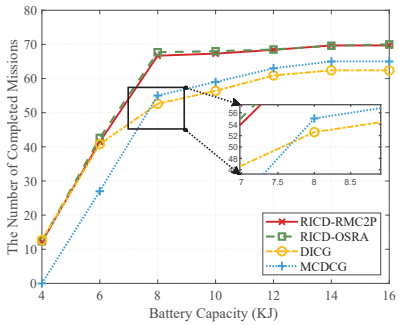

Figure D2 The number of completed missions versus the number of battery capacity.

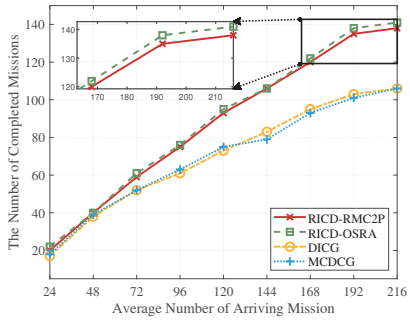

Figure D3 The number of completed missions versus the average number of arriving missions ($CB_{\max}$=40Gbits, $EB_{\max}$=8KJ, $\mathcal{K}$=4).

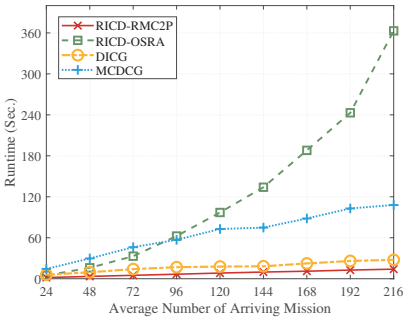

Figure D4 Runtime of resource scheduling versus the average number of arriving missions ($CB_{\max}$=40Gbits, $EB_{\max}$=8KJ, $\mathcal{K}$=4).

Appendix D.3 Performance Evaluation

In the following experiments, we investigate the performance of different algorithms in the MDSN under various network parameters, including buffer size, computing capacity, and the number of arriving missions.

To verify the effectiveness of the proposed algorithms, we plot the network reward performance of the proposed RICD-RMC2P, DQN-RMC2P, DICG, and MCDCG algorithms for each episode during the training period, as shown in Fig. D1. The network rewards of the two learning algorithms, RICD-RMC2P and DQN-RMC2P, gradually stabilize, indicating that the respective optimal cross-domain resource scheduling strategies have been learned. Moreover, the RICD-RMC2P and DQN-RMC2P strategies outperform DICG and MCDCG. This can be attributed to the foresighted nature of RICD-RMC2P and DQN-RMC2P, which enables them to maximize the long-term network rewards. In addition, RICD-RMC2P achieves higher rewards compared to DQN-RMC2P because it effectively learns the environment during training, allowing it to discern which states of the MDSN are valuable and which scheduling decisions are optimal.

In Fig. D2, we investigate the effect of the satellite battery capacity on the number of completed missions achieved by different algorithms. With $CB_{\max} = 40$ Gbits, $\mathcal{K} = 4$, and $\lambda = 96$, the percentage of incoming data in busy and idle domains is 75% and 25%, respectively. It can be observed from Fig. D2 that, as the satellite battery capacity increases, the performance of the four methods improves. This is because with more energy available, the satellite can support the reception, forwarding, and processing of more missions, thus increasing the probability of successful mission completion. The number of completed missions gradually plateaus after the energy buffer reaches $EB_{\max} = 8$ KJ. The underlying rationale is that under these circumstances, the primary limiting factor for mission completion is no longer energy, but other resources. It is worth noting that MCDCG is inferior to DICG when the energy buffer is small. However, as the energy buffer increases, MCDCG tends to trade energy consumption for more opportunities for mission completion through cross-domain scheduling, thereby outperforming DICG. RICD-RMC2P and RICD-OSRA achieve nearly identical performance, although they employ RMC2P and OSRA, respectively, for mission realization path planning. This demonstrates that our proposed RMC2P algorithm can achieve an approximation close to the exact solution. Benefiting from the joint consideration of foresighted cross-domain decision-making and efficient planning of mission realization paths, the proposed RICD-RMC2P algorithm outperforms MCDCG by 21.8% and outperforms DICG by 28.8% when the battery buffer capacity is 8 KJ.

From Fig. D3, it can be seen that as the number of arriving missions increases, the number of completed missions in the algorithm gradually increases. This can be explained by the fact that a higher arrival rate of missions enables the system to compute and transmit more missions. However, when the number of missions is large enough, the number of completed missions no longer increases. This is because in this situation, limited resources have become the bottleneck for mission completion, resulting in the inability to complete additional missions. Moreover, we can observe that as the number of mission arrivals λ increases, the difference between the number of missions completed by MCDCG and that achieved by DICG first increases and then decreases. Finally, DICG outperforms MCDCG when the number of missions increases to a certain value. This is expected, as a low value of λ implies that each domain has

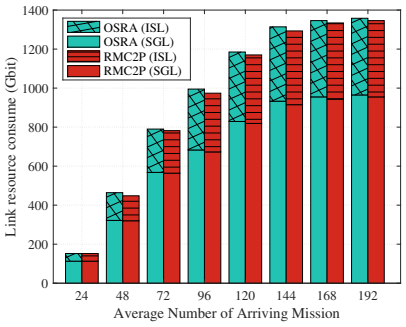

Figure D5 Communication resource consumption versus the average number of arriving missions ($CB_{\max}$=40Gbits, $EB_{\max}$=8KJ, $\mathcal{K}$=4).

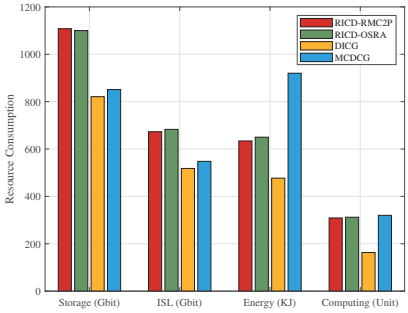

Figure D6 Resource consumption of different schemes ($CB_{\max}$=40Gbits, $EB_{\max}$=8KJ, $\mathcal{K}$=4, λ=96).

abundant resources, resulting in an insignificant performance gap in the number of completed missions between MCDCG and DICG. Nevertheless, the performance gap becomes more pronounced as λ increases since MCDCG can collaborate resources across multiple domains. However, when λ increases to a certain value, the number of completed missions of MCDCG is lower than that of DICG because 1) MCDCG requires additional energy and computing resource consumption, and 2) the resources become the main bottleneck that limits the number of completed missions. The proposed RICD-RMC2P can improve mission completion by flexibly performing cross-domain scheduling according to the real-time network state.

We investigate the effect of the average number of arrived missions on the runtime of the four methods, as shown in Fig. D4. We can notice that as λ increases, the runtime of each method increases. This is reasonable since the size of the variables to be calculated increases as the number of arrived missions increases. Table D2 summarizes the comparison of complexity. Combined with Fig. D5, it can be seen that the proposed RICD-RMC2P significantly reduces the runtime while achieving a high mission completion rate close to that of RICD-OSRA. At an average number of 192 arrived missions, the proposed algorithm reduces runtime by 96.1% compared to RICD-OSRA, 87% compared to MCDCG, and 49.4% compared to DICG. We attribute this to the efficiency of cross-domain guidance in the upper layer and the lightweight computation of the proposed heuristic algorithm in the lower layer, which is capable of solving the mission-specific realization paths quickly when the number of missions is large.

To further investigate the relationship between mission completion gain and resource utilization, Fig. D6 depicts the consumption of computing, storage, communication, and energy resources resulting from different methods. It can be seen that the proposed method is close to the OSRA in terms of multidimensional resource consumption. DICG exhibits the lowest resource consumption due to its non-sharing of inter-domain resources, which leads to resource islanding. In addition, combined with Fig. D3, we can find that the computing resource consumption of the MCDCG algorithm is approximately the same as that of the proposed method. However, the energy consumption of MCDCG is higher, and its mission completion rate is lower than that achieved by the proposed method. This is because the cross-domain scheduling can collaborate with the inter-domain resources, but requires additional energy and computing consumption for cross-domain data processing. The MCDCG's greedy cross-domain scheduling at each time slot results in additional computing and energy consumption, which means it ignores the impact of current resource usage on future long-term resource scheduling.

Table D2 Comparison of Complexity

Method	Complexity
OSRA	$\mathcal{O}(V \cdot L \cdot 2^V)$
RMC2P	$\mathcal{O}(V' \cdot L' \cdot 2^{V'} \cdot \lvert\mathcal{M}_t\rvert)$
CGRA	$\mathcal{O}(V'' \cdot L'' \cdot 2^{V''} \cdot \mathcal{I})$ ($\mathcal{I}$ iterations)

Furthermore, we investigate the effect of the variance of mission arrivals in MDSN on the number of completed missions. We define χ as the variance of the number of mission arrivals per time slot during the scheduling period. In Fig. D7, it can be seen that with the increase of χ, the number of completed missions of all methods shows a decreasing trend. Among them, the increase of χ has the least impact on the number of completed missions caused by the proposed RICD-RMC2P algorithm. We attribute this to the ability of the proposed algorithm to dynamically perform cross-domain scheduling based on the resource and mission states of multiple domains at each time slot.

We further study the effect of the number of domains in MDSN. We define μ as the value of the number of completed missions divided by the number of domains, i.e., the average number of completed missions per domain. In addition, ν represents the average number of completed missions per satellite. In Fig. D8, scene A: $\{d_1, d_2\}$ contains the domains of d_1 and d_2. Similarly, we set scene B: $\{d_1, d_2, d_3\}$, scene C: $\{d_1, d_2, d_3, d_4\}$, scene D: $\{d_1, d_2, d_3, d_4, d_5\}$, scene E: $\{d_1, d_2, d_3, d_4, d_5, d_6\}$. From Fig. D8, it can be seen that as the number of domains in the scene increases, the metrics μ and ν of DICG gradually surpass those of MCDCG. This is because the short-sighted cross-domain co-scheduling of MCDCG initially enhances mission completion as the overall size of the MDSN increases. However, the additional cross-domain energy consumption and the limited energy resources of the satellites ultimately result in lower μ and ν for the cross-domain approach compared to DICG. Furthermore, in scenarios with different numbers of domains, the metrics μ and ν of the proposed algorithm consistently outperform those of the comparison algorithms. This can be attributed to the proposed algorithm's ability to efficiently and synergistically utilize the resources of multiple domains through intelligent and dynamic cross-domain decision-making, thereby selecting the optimal cross-domain time slots to accomplish more missions.

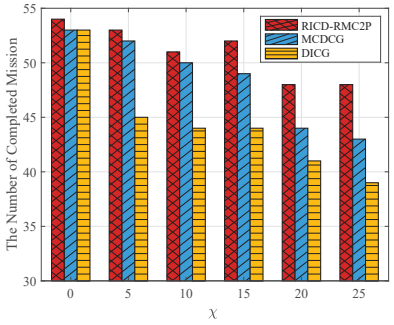

Figure D7 The number of completed missions versus the variance of mission arrivals χ. ($CB_{\max}$=20Gbits, $EB_{\max}$=8KJ, $\mathcal{K}$=4, λ=96).

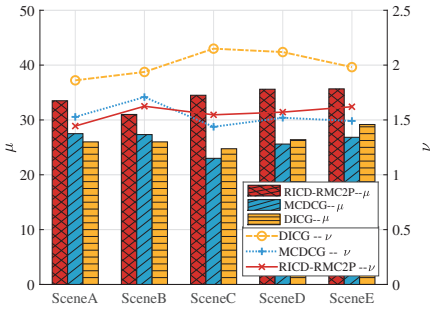

Figure D8 The number of completed missions versus the different scenes. ($CB_{\max}$=40Gbits, $EB_{\max}$=8KJ, $\mathcal{K}$=4).

Appendix E Conclusion

In this paper, we investigated the RT-CDRS problem in MDSN to effectively improve the response speed and mission completion of dynamic cross-domain scheduling. Specifically, we proposed a foresighted real-time hierarchical resource scheduling framework for MDSN. In the upper layer, we proposed the RICD algorithm, which makes real-time and flexible decisions on cross-domain resource scheduling strategies based on the state of each domain within the MDSN. This algorithm aims to optimize the overall mission completion throughout the entire scheduling period. Subsequently, to rapidly schedule a specific mission realization path in the lower layer and minimize resource cost, we proposed the RMC2P algorithm, which solves the mission realization path with low complexity. Numerical experiments validate the superiority of the proposed approach over the benchmark approaches. In addition, we also investigated and analyzed the impact of network parameters (e.g., number of domains and resources, etc.) on network performance, providing valuable insights into the important yet underexplored field of MDSN.

While the proposed framework demonstrates significant advantages, some aspects could benefit from further exploration. The reinforcement learning model requires substantial data and training time, and convergence is not always guaranteed. Additionally, the effectiveness of the hierarchical framework is influenced by the performance of the lower-layer path planner. The framework may also face challenges in adapting to significant changes in network topology, particularly given the assumption of a fixed topology per slot. Future work could focus on improving training efficiency, enhancing adaptability to dynamic environments, and refining the decision-making process to address these aspects.

References

1 Mao B, Zhou X, Liu J, Kato N. On an intelligent hierarchical routing strategy for ultra-dense free space optical low earth orbit satellite networks. IEEE Journal on Selected Areas in Communications, 2024, 42(5): 1219–1230.
2 He H, Zhou D, Sheng M, Li J. Hierarchical cross-domain satellite resource management: An intelligent collaboration perspective. IEEE Transactions on Communications, 2023, 71(4): 2201–2215.
3 Kim T, Kwak J, Choi J P. Satellite edge computing architecture and network slice scheduling for IoT support. IEEE Internet of Things Journal, 2022, 9(16): 14938–14951.
4 Zeng G, Zhan Y, Xie H, Jiang C. Networked satellite telemetry resource allocation for mega constellations. In: Proceedings of the IEEE International Conference on Communications (ICC), 2022: 4583–4588.
5 Jia Z, Sheng M, Li J, Zhou D, Han Z. VNF-based service provision in software defined LEO satellite networks. IEEE Transactions on Wireless Communications, 2021, 20(9): 6139–6153.
6 Gang Y, Zhang Y, Wu P, Zheng H, Fan G. A study for inter-satellite cooperative computation offloading in LEO satellite networks. China Communications, 2025, 22(2): 12-25.
7 He L, Li S, Jia Z, Wang J, Han Z. Joint data compression and task scheduling for LEO satellite networks. IEEE Transactions on Vehicular Technology, 2025, 74(9): 14991-14
8 Ou J, Xing L, Yao F, Li M, Lv J, He Y, Song Y, Wu J, Zhang G. Deep reinforcement learning method for satellite range scheduling problem. Swarm and Evolutionary Computation, 2023, 77: 101233.
9 Rodrigues T K, Kato N. Network slicing with centralized and distributed reinforcement learning for combined satellite/ground networks in a 6G environment. IEEE Wireless Communications, 2022, 29(1): 104–110.
10 Xu X, Wang Q, Li S, Xu H, Zhao H, Han Z. An adaptive dual-mode task-oriented resource management strategy for GEO relay systems. IEEE Transactions on Mobile Computing, 2023, 23(5): 4303–4317.
11 Fan H, Long J, Liu L, Yang Z. Dynamic digital twin and online scheduling for contact window resources in satellite network. IEEE Transactions on Industrial Informatics, 2023, 19(5): 7217–7227.
12 Chai F, Zhang Q, Yao H, Xin X, Gao R, Guizani M. Joint multi-task offloading and resource allocation for mobile edge computing systems in satellite IoT. IEEE Transactions on Vehicular Technology, 2023, 72(6): 7783–7795.
13 Lyu Y, Hu H, Fan R, Liu Z, An J, Mao S. Dynamic routing for integrated satellite-terrestrial networks: A constrained multi-agent reinforcement learning approach. IEEE Journal on Selected Areas in Communications, 2024, 42(5): 1204–1218.
14 Shen Z, Jin J, Tan C, Tagami A, Wang S, Li Q, Zheng Q, Yuan J. A survey of next-generation computing technologies in space-air-ground integrated networks. ACM Computing Surveys, 2023, 56(1): 23.
15 Yu S, Chen X, Zhou Z, Gong X, Wu D. When deep reinforcement learning meets federated learning: Intelligent multitimescale resource management for multiaccess edge computing in 5G ultradense network. IEEE Internet of Things Journal, 2021, 8(4): 2238–2251.
16 Zhou D, Sheng M, Bao C, Wang Y, Li J, Han Z. Mission-driven resource scheduling in satellite-terrestrial networks: From perspective of collaboration and reconfiguration. IEEE Transactions on Communications, 2025, Early access.
17 Pelton J N, Madry S, Camacho-Lara S. Handbook of satellite applications. Springer, 2017.
18 Liu Q, Xia T, Cheng L, van Eijk M, Ozcelebi T, Mao Y. Deep reinforcement learning for load-balancing aware network control in IoT edge systems. IEEE Transactions on Parallel and Distributed Systems, 2022, 33(6): 1491-1502.
19 Van Hasselt H, Guez A, Silver D. Deep reinforcement learning with double Q-learning. In: Proceedings of the AAAI Conference on Artificial Intelligence, 2016, 30(1).

20 Wang Z, Schaul T, Hessel M, Hasselt H, Lanctot M, Freitas N. Dueling network architectures for deep reinforcement learning. In: Proceedings of the International Conference on Machine Learning, 2016: 1995-2003.
21 Sewak M, Sewak M. Deep Q Network (DQN), Double DQN, and Dueling DQN: A step towards general artificial intelligence. In: Deep Reinforcement Learning: Frontiers of Artificial Intelligence, Springer, 2019: 95-108.
22 Ji Y, Wang Y, Zhao H, Gui G, Gacanin H, Sari H, Adachi F. Multi-agent reinforcement learning resources allocation method using dueling double deep Q-network in vehicular networks. IEEE Transactions on Vehicular Technology, 2023, 72(10): 13447-13460.
23 He H, Zhou D, Sheng M, Li J, Chau Y. Intelligent Collaborative Scheduling Enabled Communication-Computing Integration in Multi-Layer Satellite Networks. IEEE Transactions on Communications, 2025, 73(11): 10778-10793.
24 3GPP TS 23.401, V19.1.0. GPRS enhancements for Evolved Universal Terrestrial Radio Access Network (Release 19), Dec. 2024.
25 New use case on remote sensing in satellite. Available: https://www.3gpp.org/ftp/tsg_sa/WG1_Serv/TSGS1_106_Jeju/Docs/S1-241131.zip. Accessed: Mar. 10, 2025.