Research Article

Space Information Network Resource Scheduling for Cloud Computing: A Deep Reinforcement Learning Approach

Yufei Wang,1 Jun Liu,1 Yanhua Yin,2 Yu Tong,1 and Jiansheng Liu3

1School of Computer Science and Engineering, Northeastern University, Shenyang 110169, China
2Meteorological Information Center of Liaoning Province, Shenyang 110166, China
3Neusoft Research, Northeastern University, Shenyang 110169, China

Correspondence should be addressed to Jun Liu; liujun@cse.neu.edu.cn

Received 21 September 2021; Revised 26 January 2022; Accepted 11 February 2022; Published 9 March 2022

Academic Editor: Huaming Wu

Copyright © 2022 Yufei Wang et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

With the development of satellite technology, space information networks (SINs) have been applied to various fields. SINs can provide more and more complex services and receive more and more tasks. The existing resource scheduling algorithms are difficult to play an efficient role in such a complex environment of resources and tasks. We propose a resource allocation scheme based on reinforcement learning. Firstly, according to the characteristics of the resources of SINs, we established the cloud computing architecture of SINs to manage the resources uniformly. Next, we adopt a variable granularity resources clustering algorithm based on fuzzy and hierarchical clustering algorithms. This algorithm can adaptively adjust the resource size and reduce the scheduling range. Finally, we model the resource scheduling process by a reinforcement learning algorithm to solve the joint resource scheduling problem. The simulation results show that the scheme can effectively reduce resources consumption, shorten the task execution time, and improve the resource utilization efficiency of SINs.

1. Introduction

As an essential platform for all kinds of space applications, SINs have the advantage of global coverage. It can provide meteorological, positioning, and communication services and is outstanding in navigation, rescue, and remote sensing applications. SINs can provide collaborative management for all kinds of space equipment and improve the utilization efficiency of space equipment, which has been widely concerned by researchers [1–3]. After years of SIN development, all kinds of spacecraft have made significant progress, constantly improving the load capacity of resources and enriching all types of tasks. Although the diversity of mission execution has been dramatically improved, it is difficult to quickly break with the limited resources on the satellites. Therefore, with the continuous development of technology and the increasing number of tasks, the resource allocation ability of SINs needs to be increased. The current resource allocation technology and network architecture are difficult to support the resource allocation of SINs with more and more complex tasks [4].

SINs need innovative network architecture and scheduling scheme. In recent years, researchers have done a lot of work on the architecture of SINs. Yang et al. applied software defined network (SDN) to satellite networks to solve the problems of isolation and private use of satellite network resources. They designed software defined satellite network (SDSN) architecture [5]. Zhang et al. proposed a software defined air-ground integrated network architecture to enable network resources to switch freely [6]. In addition to applying SDN to SINs, edge computing is also concerned by researchers. Zhu et al. proposed the edge computing architecture of satellites to unload computing tasks to visible satellites for execution [7]. Wang et al. solve the capacity limitation of resources by applying the new mobile edge computing of satellite network architecture [8]. There are
few studies on eliminating resource heterogeneity and the sharing architecture between resources in the above studies. Due to the limited and dynamic resources of SINs, a new architecture is needed to share the resources of SINs. Cloud computing is worthy of our reference. This paper proposes the cloud computing architecture of SINs.

In addition to the network architecture, there has been some critical research on the resource scheduling algorithm of SINs. Early researchers mainly applied dynamic programming and search algorithms to model to complete satellite resource scheduling. Bianchessi et al. studied the management of multiple satellites through a tabu search heuristic algorithm [9], Liu et al. modeled the multisatellite scheduling problem as a dynamic constraint problem and solved the search algorithm for the new tasks [10]. Habet et al. modeled satellite scheduling as a hybrid system search and tabu search algorithm and constrained by visible time window [11]. Lemaître et al. studied satellite resource scheduling through a dynamic programming algorithm and analyzed the advantages of the proposed algorithm by comparing greedy algorithms [12]. Wang et al. designed a scheduling decision system to manage satellite resources through a nonlinear model and applied priority heuristic ideas and conflict avoidance in the system [13]. Dilkina and Havens improved the simulated annealing algorithm, spread the observation time window of satellite resource scheduling to the local search process, and carried out satellite resource scheduling through a model annealing algorithm [14]. Li et al. research on satellite resource scheduling through combinatorial genetic algorithm [15]. Geng et al. applied the genetic algorithm to satellite resource scheduling by improving the coding mode of the genetic algorithm. They improved the genetic algorithm by using the mixed coding mode of binary and integer coding [16]. Marinelli et al. studied the scheduling problem of satellite resources through a heuristic prime search algorithm [17, 18].

There are many single objective optimization algorithms in the above algorithms. With the increasing business of SINs, satellite resource scheduling needs to consider multiple objectives. Many researchers have studied the multiobjective satellite resource scheduling algorithm. Tangpattanakul et al. optimize the satellite user demand target as a local search and optimize the total target revenue and user fairness in the algorithm design [19]. Sun et al. applied four objectives to complete task planning during optimization so that the scheduling results can better meet users’ needs [20]. In Li et al.’s study, when designing the algorithm, multiple objectives such as profit and the number of observation tasks are considered. A coding and decoding method is proposed using heuristic rules. The results show that the performance of this method is remarkable [21].

Heuristic algorithms are often used in multiobjective optimization in the above research, and these algorithms have several problems. The operator and parameter setting in the heuristic algorithm depend on expert experience, and the results are pretty different when the parameters are different [22]. With the increasing complexity of the environment and the increase of constraint problems, the search process of the heuristic algorithm will increase sharply, and the difficulty of algorithm solutions will increase [23]. Recently, reinforcement learning, an essential branch of machine learning, can better analyze dynamic decision-making problems, which provides ideas for solving the above issues. At present, there are many reinforcement learning algorithms, such as Deep Q-Learning Network (DQN) [24] and Deep Deterministic Policy Gradient (DDPG) [25]. Reinforcement learning algorithm has been widely used in resource allocation in the Internet of things and cloud environment [26–30]. Rjoub et al. proposed a variety of scheduling algorithms based on reinforcement learning for cloud computing resource scheduling. Experiments show that deep reinforcement learning combined with the LSTM (Long Short-Term Memory) algorithm has good performance [31]. Arisdakessian et al. studied the integration of cloud computing resources in the IoT and proposed a multistandard intelligent scheduling method using game theory, which has good ability in service execution time and resource integration [32]. Cheng et al. studied the impact of resource supply and task scheduling on energy in cloud computing and proposed a DRL-Cloud algorithm to effectively reduce the energy cost [33]. In recent years, reinforcement learning algorithm has attracted much attention in the research of resource scheduling in SINs. Qiu et al. jointly manage the satellite network’s network, cache, and computing resources through the deep reinforcement learning method [34]. Deng et al. applied deep reinforcement learning to model the allocation process of satellite network resources, showing the specific function of satellite resources allocation through deep reinforcement learning [35]. Liao et al. proposed a cooperative multiagent deep reinforcement learning framework to realize the management of wireless resources in satellite networks [36]. In addition, other researchers have applied reinforcement learning algorithms to satellite network resource scheduling [37–40].

The reinforcement learning algorithm can overcome the shortcomings of the heuristic algorithm. Still, in most studies of SINs, there are few multiobjective optimization studies to reduce the use of resources directly. The reinforcement learning algorithm cannot overcome the shortcomings of the heuristic algorithm. Still, in most studies of SINs, there are a few multiobjective optimization studies to reduce the use of resources directly. At the same time, there is less research on the diversity of SIN resources. Because SINs contain a variety of resources, dividing and matching a variety of resources in detail are also a problem to be considered in SIN resource scheduling. To solve the above issues, this paper proposes DQN Resource Scheduling (DQNRS) algorithm. Based on the DQN algorithm, this algorithm makes a detailed analysis of SIN resources and focuses on the use and types of SIN resources. The main contributions of this paper are summarized as follows:

(1) We designed the cloud computing architecture of SINs and divided the SINs into space service facilities, space IoT facilities, and ground service facilities. SIN architecture is the first step of space resource scheduling. A perfect SIN architecture can improve the efficiency of resource scheduling.
(2) We perform adaptive variable granularity clustering on infrastructure layer resources in the space cloud environment. Firstly, we aggregate SIN resources into resource particles with different attributes through fuzzy clustering. Next, we determine the resource granularity through hierarchical clustering. This method can reduce the search range of resources and solve inaccurate matching in resource scheduling caused by the diversity of SINs.

(3) We design an intelligent scheduling algorithm for the management layer of the space cloud environment. According to the characteristics of infrastructure layer resources and application layer tasks, we establish constraints such as time consumption model, resource occupation model, and location coverage model. We redesigned the state, action, and reward of the DQN algorithm according to the constraints; improved the action exploration strategy; and matched the tasks to the appropriate resources through the DQN algorithm.

(4) Experimental results show that the DQfNRS algorithm proposed in this paper performs better than the ACO algorithm and random and FCFO algorithms commonly used in cloud computing regarding resource usage and task execution time.

2. System Description and Model

2.1. Space Information Network Resource Cloud Computing Architecture. In the traditional satellite architecture, due to the insufficient processing capacity on the satellite, the data information obtained by the satellite can only be transmitted to the ground for processing and forwarded to users. There are some problems in dynamic space networks, such as waiting for the return. To solve the above issues, we designed the SIN cloud computing architecture. The architecture can share the resources of SINs and realize task processing in various situations. As shown in Figure 1, the left half of Figure 1 is the infrastructure part of the cloud computing architecture, and the right half is the layered function of the cloud computing architecture. The architecture realizes the efficient utilization of SIN resources through the cloud data center. It applies to cloud computing ideas to integrate heterogeneous computing resources, storage resources, and perceptual resources in SINs. It can realize the resource sharing and collaborative service of SINs.

The cloud computing infrastructure layer of SINs includes high orbit satellite networks based on geosynchronous satellites, near-earth satellite networks of medium and low orbit satellites, and ground servers. As shown in Figure 1, it is divided into space service facilities, space IoT facilities, and ground service facilities. The space service facility consists of high orbit satellites, including multiple geosynchronous satellite clusters. It can always be connected with near-earth satellites and ground servers. It can provide various types of services and can analyze and process tasks without returning to the ground. It is the space service center of cloud computing architecture. Space IoT facilities are composed of near-earth satellites, including all kinds of satellites flying around the Earth. It can provide high-quality earth observation services, collect various signals in real-time, and transmit information and data to space or ground servers. The ground service terminal is composed of several ground service centers. The service center can carry out large-scale computing and storage. When the service capacity of the space service terminal is insufficient, it can transfer the tasks back to the ground service facilities. It can quickly perform various tasks of the SINs. It is the ground service center of cloud computing architecture. Finally, the hardware facilities of the infrastructure layer are virtualized to generate a virtual resource pool for scheduling by the cloud computing management layer to prepare for scheduling by the management layer. For the infrastructure of space networks, SDN technology is used to manage the space-dispersed satellites’ physical nodes. This method realizes the data processing platform of “physically distributed and logically unified.” The satellite server nodes and the general control center are virtualized by virtualization technology to provide services such as computing, storage, and data forwarding for service requestors. We can realize the functions of collaborative management, flexible scheduling, and on-demand expansion of resources through virtualization technology.

The cloud computing management layer of SINs can provide functions such as resource monitoring, task receiving, and task decision-making distribution. This layer can access the resource pool of the infrastructure layer. At the same time, this layer can provide services for the application layer, including navigation, communication, and remote sensing. After the application layer user submits the task requirements, the management layer makes decisions through the scheduling algorithm according to the task requirements and resource environment. It sends the tasks to various resources in the resource pool.

The cloud computing application layer of SINs is a service that the management can provide through resource scheduling decisions. Relying on the facilities and data in the SINs and using the current satellite processing technology, it can provide remote sensing, navigation, and communication services. The application layer is responsible for returning the task results processed by the management layer to the users. At the same time, the application layer is responsible for receiving the requirements of the tasks submitted by users and sending the task requirements to the management layer for decision algorithm resource scheduling.

The cloud computing architecture of SINs can share all kinds of resources, realize the interconnection between resources, realize large-scale task processing through the cooperation between resources, and realize the operation of data processing, acquisition, and transmission without returning information to the ground.

2.2. Space Cloud Environment Infrastructure Layer Resource Processing. The infrastructure in the space cloud environment includes a variety and many resources, and the scale of resources is complex. However, the task requirements in
the space cloud environment are very changeable, and a single resource dimension is difficult to meet the task requirements. The resource forms with different granularity are an effective way to execute tasks quickly. We propose a variable granularity adaptive clustering algorithm. Firstly, the particles are obtained through the clustering algorithm, then, the particles are transformed through the aggregation network, and finally, the appropriate particle layer and clustering results are obtained.

In SINs, resources are usually heterogeneous, and different resources can provide various capabilities, such as computing, storage, communication, and perception. Most scheduling studies do not distinguish between resource capabilities, increasing the cost of scheduling, and there may be unreasonable scheduling. These problems will have a specific impact on the completion time of the tasks and the utilization of resources. In SINs, some tasks may have a large amount of data, but the amount of calculation of these tasks is tiny, such as shooting image storage. However, some other tasks may require a lot of computing, such as various decision tasks, and some tasks may not require extensive data and computing but need real time, such as the display of some video data. Therefore, we use a variable granularity clustering algorithm to cluster the resources in SINs, and we need to consider the characteristics of SIN resources fully. We distinguish resources as follows: computing resources, storage resources, communication resources, navigation resources, and perception resources. These resource attributes can abstractly represent various resources’ functions and jointly define the resource capabilities of the space cloud environment.

The resources model of SINs can be represented by $R = \{ r_{\text{cal}}, r_{\text{stor}}, r_{\text{tran}}, r_{\text{nav}}, r_{\text{per}} \}$, where $r_{\text{cal}}$ represents the computing power of the resource, which is calculated by the number of instructions executed per second; $r_{\text{stor}}$ represents the storage capacity of resources, which is calculated by the storage capacity of resources; $r_{\text{tran}}$ represents the transmission capacity of resources, which is calculated by the bandwidth of resources; $r_{\text{nav}}$ represents the navigation capability of the resource, which is calculated by the coverage of the resource; and $r_{\text{per}}$ represents the perception ability of resources, which is calculated by imaging separation rate.

Multiple resources can form a resource cluster, which is represented by $R_{\text{total}} = \{ R_1, R_2, \ldots, R_n \}$. A resource cluster is a collection of virtual resources in a virtual resource pool. The result of resource clustering is to find different levels of virtual sets, namely, resource clusters, through a clustering algorithm. The granularity of each virtual resource set is different, which can provide accurate services for different task requirements. The scheduling algorithm puts forward additional requests for resources with different granularity according to the needs of tasks, which can effectively reduce the resource search space and speed up the speed of resource scheduling.

Resource variable granularity clustering algorithm is completed by secondary clustering. The first clustering is fine-grained clustering, which is used to analyze the internal characteristics and structure of resources in detail—the different results obtained by the first clustering complete the solution of particles in the aggregation network. Next, granular clustering is completed. Granular clustering is mainly the aggregation of particles under different similarity thresholds. The similarity threshold is continuously adjusted to find the appropriate clustering particles and particle layers in the clustering process. If the clustering particles are too fine, the aggregation particle layer will be dense and slow.
scheduling time. If the clustering particle is too coarse, the aggregation layer is sparse, and the scheduling is inaccurate.

After determining the resource model and characteristics, according to variable granularity clustering, the resource clustering of SINs is divided into the following stages: data standardization, resource classification, cluster particle generation, and cluster particle layer generation. The clustering process is discussed in the steps below.

2.2.1. Establishing Data Matrix. According to the five attributes of the resource attribute set \( R = \{ r_{\text{cal}}, r_{\text{stor}}, r_{\text{tran}}, r_{\text{nav}}, r_{\text{per}} \} \), the original data matrix of SIN resources can be listed as \( R = \{ R_i | R_i = \{ r_{\text{cal}}^i, r_{\text{stor}}^i, r_{\text{tran}}^i, r_{\text{nav}}^i, r_{\text{per}}^i \}, 1 \leq i \leq n \} \), expressed as

\[
R = \begin{bmatrix}
  r_{\text{cal}} & r_{\text{stor}} & r_{\text{tran}} & r_{\text{nav}} & r_{\text{per}} \\
  r_{\text{cal}} & r_{\text{stor}} & r_{\text{tran}} & r_{\text{nav}} & r_{\text{per}} \\
  \vdots & \vdots & \vdots & \vdots & \vdots \\
  r_{\text{cal}}^{n-1} & r_{\text{stor}}^{n-1} & r_{\text{tran}}^{n-1} & r_{\text{nav}}^{n-1} & r_{\text{per}}^{n-1} \\
  r_{\text{cal}}^n & r_{\text{stor}}^n & r_{\text{tran}}^n & r_{\text{nav}}^n & r_{\text{per}}^n
\end{bmatrix}.
\]

2.2.2. Data Standardization. In SIN resources, the orders of magnitude of various resources are different. Suppose you want all the quantities to be calculated and compared together. It is necessary to normalize the values of various resources to a certain range, eliminating the differences caused by the significant differences of a certain type of values.

The first step is to standardize the data and calculate the average value and standard deviation of resource capacity. Through this operation, standardized data \( r'_i \) can be obtained, as shown in

\[
r'_i = \frac{r_i - \bar{r}_k}{S_k},
\]

In the formula, \( \bar{r}_k = \frac{1}{n} \sum_{i=1}^{n} r_{ik} \) is the average value of the \( k \)th attribute of the resource, \( S_k = \sqrt{\frac{1}{n} \sum_{i=1}^{n} (r_{ik} - \bar{r}_k)^2} \) is the standard deviation of the resource attribute. In addition to the above solution, dimensional elimination calculation is also required. The calculation formula is shown in

\[
r''_i = \frac{r''_i}{\max (r''_i) - \min (r''_i)}.
\]

2.2.3. Resource Classification. Calculate the proportion of each attribute capability value of each resource, and get the attribute capability with the highest proportion of the resource according to the proportion value. Then, the resource category is this kind of resource. The classification formula (4) is expressed as

\[
C_{ik} = \max \left\{ \frac{r_{i1}}{\sum_{k=1}^{m} r'_{ik}}, \frac{r_{i2}}{\sum_{k=1}^{m} r'_{ik}}, \ldots, \frac{r_{im}}{\sum_{k=1}^{m} r'_{ik}} \right\}.
\]

The resource category is obtained according to \( C_{ik} \). The most extensive resource attribute capability in each category is the category of the resource. Firstly, the resources of a resource category are divided into a resource cluster. Similarly, \( m \) resource clusters can be obtained. Finally, each resource cluster will be fuzzy clustered.

2.2.4. Cluster Particle Generation. A fuzzy matrix calculates particles. In the first step, the fuzzy similarity matrix needs to be calculated and measured by the similarity between resources. The similarity of resources \( R_i \) and \( R_j \) is expressed as \( p_{ij} \). We use the exponential similarity coefficient method to calculate the similarity, so we can use the similarity to construct the fuzzy similarity matrix. The calculation method \( p_{ij} \) is shown in

\[
p_{ij} = \frac{1}{m} \sum_{k=1}^{m} \exp \left( -\frac{3(\hat{r}_{ik}' - \hat{r}_{jk}')}{4S_k} \right),
\]

where \( m \) is the number of resource attribute capability indicators and \( \hat{r}_{ik}' \) is the \( k \)-dimensional performance indicator of the resource \( r_i \). We can construct the fuzzy similarity matrix \( P = (p_{ij})_{mn \times mn} \) between resources according to \( r_{ik}' \).

Transitivity is necessary when solving clustering results, but the fuzzy similarity matrix does not have this property. It is essential to calculate the fuzzy equivalent matrix further and solve it by the transitive closure method based on the fuzzy similarity matrix. This solution process \( P' \) is obtained by circularly calculating the similarity matrix \( P \), and \( r_{ij} \) does not change at the end of the calculation to get the fuzzy equivalent matrix \( P' \). The solution method of transitive closure is shown in

\[
r_{ij} = r_{ij} = \frac{m}{k=1} (r_{ik} \wedge r_{jk}) = \max \{ r_{ik} \wedge r_{jk} \}, \quad 1 \leq k \leq m.
\]

Different thresholds can obtain other clusters. At this time, particles are obtained. The performance of particles is different, but the difference is slight.

2.2.5. Cluster Granular Generation. The obtained particles are hierarchically clustered, and the particle layer is obtained by average group proximity. Single join calculates the distance between the two most similar samples in each pair of clusters and merges the clusters of the two nearest samples. For a cluster composed of one or more samples in each particle, the single connection calculation method finds the minimum distance between each cluster sample and merges two clusters.

The resource cluster obtained by variable granularity clustering can be expressed as \( R_{\text{layer}} = \{ R_{i1}', R_{i2}', \ldots, R_{im}' \} \), where \( i \) represents the granular layer of resources after clustering, \( j \) represents the resource attributes after classification in resource clustering, and \( m \) represents the number of resource clusters under this granular layer.
According to the analysis of variable granularity clustering of resources, the clustering algorithm is designed in detail, as shown in Algorithm 1. Firstly, the fuzzy clustering algorithm is used to cluster the resource dataset for the first time to obtain the cluster particles. Then, the hierarchical clustering algorithm is used to cluster the dataset for the second time to form the resource particle layer. If the particle is larger than the task demand, the particle clustering is not satisfactory enough. Repeat particle clustering until the mean value of particles is less than the task demand.

2.3. Space Cloud Environment Management Layer Resource Scheduling Algorithm. After the infrastructure layer resources of the space cloud environment are processed, the tasks to resource matching decisions need to be completed at the management layer of the space cloud environment. To carry out efficient resource scheduling, in this section, we design a resource scheduling algorithm suitable for the space cloud environment based on DQN under the constraints of task execution time, resource usage, and location coverage in SINs. This section describes the resource scheduling of the SINs as after the cloud environment application layer receives the task, it decides the management layer, which infrastructure layer resources will be used to perform the task. Our goal is to overlap the task demand location and resource coverage location, minimize the execution time of the task, and occupy all kinds of resources in the information network as little as possible.

2.3.1. Problem Constraints. According to the resource scheduling characteristics of SINs, the constraints of the model are as follows.

1. **Time Consumption Model.** The total time required for the task \( t_j \) to execute the process on the virtual resource cluster \( R_j \) is expressed in \( \text{totalTime}_{ij} \), where \( t_j \) and \( R_j \) are the comprehensive representation of task and resource modeling, respectively. The total time \( \text{totalTime}_{ij} \) consists of the waiting time \( \text{waitTime}_{ij} \) before task execution and \( \text{workTime}_{ij} \) during task execution, which is expressed by

\[
\text{totalTime}_{ij} = \text{waitTime}_{ij} + \text{workTime}_{ij},
\]

where \( \text{waitTime}_{ij} \) represents the time that the task needs to wait for the resource cluster before executing on the resource cluster and \( \text{workTime}_{ij} \) is the execution time after the task is deployed on the resource cluster, expressed by

\[
\text{workTime}_{ij} = \begin{cases} 
\text{worktime}, & \text{worktime} \neq 0, \\
\frac{t_{\text{need}}}{R_{\text{ability}}}, & \text{worktime} = 0.
\end{cases}
\]

\( t_{\text{worktime}} \) represents the execution time attribute in the task. If the task is continuous, the execution time of the task is directly equal to the time. If the task is timely, the execution time of the task needs to be calculated. The execution time is calculated by the resource amount \( T_{\text{need}} \) required by the task and the resource amount \( R_{\text{ability}} \) that the resource can provide.

2. **Resource Occupancy Model.** The cluster’s resources \( R_j \) are occupied during task execution, and the resource amount of the cluster is represented by \( R^j_{\text{ability}} \). The resources occupied by the execution process can be determined by the execution time \( \text{workTime}_{ij} \) and the amount of resources \( R^j_{\text{ability}} \), so the total resource usage obtained by the task according to the deployment plan \( P \) is expressed by

\[
\text{totalUse}(P) = \sum_{j=1}^{m} \left[ R^j_{\text{ability}} \cdot \text{workTime}_{ij} \right].
\]

After one of the resource clusters is determined, the resource cluster’s ability when performing tasks is also certain, and the use of resources per unit time can be determined.

3. **Location Coverage Model.** The request area of the task \( T_i \) and the coverage area of the virtual resource cluster \( R_j \) sometimes do not coincide. We need to determine the visibility of the satellite and the ground request area. It is assumed that the Earth is a spherical body with uniform mass, the radius of the Earth is \( r_e \), the height of resource cluster \( R_j \) is \( h \), and the subsatellite point of the satellite on the Earth’s surface is \( R^j \). In the figure below, as the tangent line between the satellite and the ground (tangent point \( M \)), the earth surface area surrounded by the line tangent between the satellite and the ground surface is called the instantaneous maximum coverage area of the satellite, \( \beta = \alpha \cdot \text{ROM} \) is called the coverage angle of satellite to ground, \( \alpha = \angle \text{ROM} \) is called the field angle of view of the satellite to the ground, as shown in Figure 2.

Due to the occlusion of objects on the Earth’s surface, the effect of satellite observation and communication near the edge of the largest coverage area of the Earth’s surface is usually not very good. To improve the communication and observation effect of the satellite on the Earth’s surface area, it is stipulated that the included angle between the satellite line of sight and the tangent plane passing through \( M \) on the ground shall not be lower than a certain angle, which is called the minimum observation angle. The coverage angle corresponding to the minimum observation angle is \( \beta_{\sigma} \), and the corresponding satellite field angle is \( \alpha_{\sigma} \). The above analysis shows that the larger the minimum observation angle \( \sigma \), the smaller the corresponding \( \beta_{\sigma} \), and the better the communication and observation effect of the satellite on the Earth’s surface area.

The longitude and latitude of the point of the sub-satellite of resource \( R_j \) are \((\text{longitude}_j, \text{latitude}_j)\). The longitude and latitude of the task request area \( R_i \) are \((\text{longitude}_i, \text{latitude}_i)\). At this time, the geocentric angle
between the mission request area and the subsatellite points can be calculated by

\[
\beta_{ij} = \arccos \left( \frac{\sin (\text{latitude}_i) \cdot \sin (\text{latitude}_j)}{C_0/C_1} \right) + \cos (\text{latitude}_i) \cdot \cos (\text{latitude}_j) \cdot \cos (\text{longitude}_i - \text{longitude}_j),
\]

(10)

Thus, the smaller the value \( \beta_{ij} \), the larger the corresponding coverage area. Thus, the smaller the value \( \beta_{ij} \), the better the communication and observation effect of the corresponding coverage area. Therefore, in the process of selecting resources, the task chooses \( \beta_{ij} \) smaller resources as much as possible.

2.3.2. Resource Scheduling Algorithm based on DQN. Aiming at the corresponding problem model and resource preprocessing, we express the resource scheduling problem of SINs as a DQN process. We improve the state space, behavior space, and reward design to adapt to the resource scheduling process. Q-learning is one of the classical reinforcement learning algorithms and one of the artificial intelligence algorithms. The algorithm can store actions and states through Q-table \( (s, a) \). The expected Q value of the current state can be expressed as

\[
Q(s, a) = r + \gamma \max_a Q(s', a'),
\]

(11)

where \( r \) is the reward, which can be obtained through strategy \( \pi \), \( \gamma \) is the discount factor, and \( (s', a') \) is the next state and corresponding action, respectively.

The Q-table is constantly updated during the algorithm’s execution, and the environment state \( s \) is initialized at the beginning of the algorithm’s execution. The execution process is represented by episode. Each episode will calculate the corresponding reward \( r \) and the next
state $s'$ according to the current state $s$ and strategy $\pi$. Strategy $\pi$ is the probability of choosing which action to perform in the current state. The specific update process of the Q-table can be expressed as

$$Q(s, a) \leftarrow Q(s, a) + \beta \left( r + \gamma \max_{a'} Q(s', a') - Q(s, a) \right),$$

where $\beta \in (0, 1)$ is the learning rate. The reward $r$ is the reward value obtained after selecting an action. When the chosen action is a behavior beneficial to the environment, the reward will be positive feedback. Through the continuous training of episodes, a series of sequences of states, actions, and rewards can be obtained, stored in the Q-table, and rapid and beneficial action selection can be made at the next state change.

The resource status of SINs is complex and diverse, and the Q-table can no longer meet the needs. We use the DQN...
algorithm based on resource preprocessing to search in such a large environment and action. The algorithm uses deep learning to improve the dependence of reinforcement learning on the Q-table. The actions and states originally stored in the Q-table are predicted by the neural network. According to this method, reinforcement learning can be applied to complex state spaces. The detailed idea of the DQN algorithm is to generate action $a$ and reward value $r$ by using a neural network model and training network parameter weight vector $\theta$ and bias $b$. The input and output of the neural network are state $s$ and action values $a$, respectively. The specific process is shown in Figure 3.

After the neural network outputs the Q value, we should explore enough action space and learn as many states and action sets as possible. Researchers generally use $\epsilon$-greedy strategy for action selection, but the exploration probability of $\epsilon$-greedy strategy for all actions is equal, and a certain state of this method may not fully explore the action space. In the case of large action space and multiple suboptimal solutions, insufficient exploration may lead to the network falling into the local optimal solution and difficult to jump out. To solve the above problems, we use Boltzmann exploration [41] for action selection. This method can avoid completely random action selection, and the action selection probability is expressed as

$$p(a|s) = \frac{\exp (\beta Q(s,a))}{\sum_{a' \in A} \exp (\beta Q(s,a'))},$$

where $\beta$ is the exploration coefficient. The larger the coefficient, the smaller the exploration range. This method takes the Q value as the probability basis, which can ensure that when the Q values of multiple suboptimal solutions are close, each suboptimal solution has a high probability to be explored, and some poor choices are explored as little as possible. This paper combines $\epsilon$-greedy method and Boltzmann exploration method, namely, $\epsilon$-Boltzmann, to make action selection. We use the probability of $\epsilon$ to explore the action according to the above formula (13); otherwise, we choose the best action.

To better predict the Q value and the next action, it is necessary to continuously train the parameters of the neural network. The parameters of the neural network will be updated by backpropagation and gradient descent. The goal of DQN is to keep the Q value close to the target-Q-value.

![Figure 4: Change of task execution time with the number of tasks.](image-url)
The target-Q-value can be calculated by the Q-learning algorithm. In this paper, the mean square error is used as the loss function to update the parameters of the DQN neural network, as shown in

$$p(a|s) = \frac{\exp(\beta Q(s, a))}{\sum_{a'\in A} \exp(\beta Q(s, a'))},$$

(14)

where $r + \gamma \max_{a'} Q(s', a'; \theta^-)$ is the process of obtaining the target-Q-value using the target network, $s'$ is the state at the next time, $a'$ is the action selected when executing the greedy strategy, and the parameter $\theta^-$ is the corresponding parameter of the target network. $r + \gamma \max_{a'} Q(s', a'; \theta^-)$ is the process of obtaining the target-Q-value using the target network, $s'$ is the state at the next time, $a'$ is the action selected when executing the greedy strategy, and the parameter $\theta^-$ is the corresponding parameter of the target network. The parameter of the evaluation network is $\theta$, and the corresponding estimated value is $Q(s, a, \theta)$. As shown in Figure 3, the working process of the whole DQN continuously updates $\theta$ and $\theta^-$ through the loss function, and the training goal is to adjust the network parameters to minimize $L(\theta)$, where $\theta$ is updated at each step and $\theta^-$ is updated after multiple steps to reduce the correlation between networks.

Due to the dynamic changes of task request location and required resources, resource scheduling should consider the resource environment of the whole SINs. In this paper, a resource scheduling algorithm DQNRS based on the DQN framework is proposed to minimize task execution time and resource occupation to complete resource scheduling. The algorithm ensures that the resource scheduling requirements of SINs are fully considered through three parts.

1. **State.** The status consists of task execution time, resource consumption, and coverage angle. Therefore, the status can be expressed as $s = \{\text{TotalTime}, \text{TotalUse}, \beta\}$, and the task status can be determined by the task’s execution time, the occupation of resources, and the coverage angle, respectively.

2. **Action.** In this algorithm, the action is composed of $T$ and $R$, and the action vector can be given as $a = \{a_t', \forall t \in T$, $\forall r \in R\}$, where $a_t'$ indicates that the agent reasonably
schedules the resource $R$ according to the task $T$ for task execution.

(3) Reward. After selecting the action in each step, the agent can obtain certain benefits. Through the calculation of benefits, the algorithm can achieve the expected results to minimize the time and resource occupation in task execution. At the same time, location coverage is crucial in task execution, so the benefits also need to be considered to make the location coverage as large as possible. Therefore, these three objectives should be regarded as in the reward based on environmental feedback. We express the DRL agent’s return from the environment as

$$r = w_1 \times \frac{1}{\text{TotalTime}} + w_2 \times \frac{1}{\text{TotalUse}} + w_3 \times \frac{1}{\beta},$$  \hspace{1cm} (15)$$

where $\text{TotalTime}$ represents the total time of task execution, $\text{TotalUse}$ represents the total resources occupied during task execution, and $\beta$ represents the location coverage angle.

As shown in Figure 3 and Algorithm 2, given the state, action, and reward, we first establish the evaluation and target network of random parameters $\theta$ and $\theta^-$ and store state $s$ and action $a$ by building an experience playback pool. In the process of resource allocation for each task $k$, we first initialize the state. Then, for each step $i$, the input state $s$ of the network is evaluated, and the action $a$ is selected according to the output of the evaluation network through the $\varepsilon$-Boltzmann strategy. We get rewards $r$ and the next state $s'$ through action $a$. To reduce the correlation of samples, we store the obtained $(s, a, r, s')$ in a playback pool $D$ to update the estimated network.

The parameters of the evaluation network and the target network are updated using the loss function (1). The parameters of the target network are updated after multiple steps. Resource scheduling is aimed at making resources cover the required location of tasks and minimize task execution time and resource usage. These goals can be achieved through our predefined rewards. In the process of SIN resource scheduling, if the task demand location is closer to the resource service location or the resource usage is smaller, the agent will obtain positive feedback. With the continuous accumulation of experience, the tasks in the space cloud environment can be allocated to more appropriate resources.

3. Analysis of Experimental Results

3.1. Experimental Settings. We simulate the cloud computing architecture of SINs through STK and can obtain visibility and location information between satellites and resources.
We employ the simulation framework Simpy to build the data center and use TensorFlow to complete the design of the algorithm. The environment in which the algorithm runs consists of an Intel Core i5-9500 CPU, 8 GB Random Access Memory, Nvidia GeForce GT 730 GPU, and Windows 10.

Referring to the ICO constellation and Odyssey constellation, we established the SINs and cloud computing network of GEO, MEO, LEO, and ground stations. The GEO track height is 35786 km, and the MEO track height is 10355 km. The ground station latitudes are 45.75 deg, 29.65 deg, and 23.12 deg, respectively. The longitudes of the ground stations are 126.65 deg, 91.1 deg, and 113.25 deg, respectively. The capacity of the resources in the SINs is shown in Table 1.

We record the satellite position in one cycle of the simulation model. We divide the period into multiple time slices and disregard the change in the satellite position in each time slice.

The number of neural network layers of DQNRS is 3, and the learning rate of the reward value is $\gamma = 0.9$. When applying the strategy $\varepsilon$-Boltzmann, $\beta = 1$, we set the initial value of the probability $P_\varepsilon$ of the Boltzmann exploration to 1 and gradually decrease it at the rate $\alpha = 0.8$ during the training process. When $P_\varepsilon = 0.3$, its value remains constant. We mainly compare the latest SIN scheduling DRL-VNE algorithm [42] with the classical resource scheduling strategy, heuristic ACO algorithm [43], which is a first come first serve (FCFO) scheduling strategy [44] that is commonly employed in cloud computing scheduling, and random scheduling algorithm.

3.2. Simulation Result. Next, we will analyze the impact of tasks and resources on task execution time, resource usage, and task execution success rate. Figures 4–6 show the performance changes when the number of satellite nodes is 100 and the corresponding tasks are increased. Figures 7–9 show the performance changes when the number of satellite nodes increases and the number of tasks is 500. One satellite node can generate multiple virtual machines in the cloud environment.

Figure 4 shows the corresponding task execution time after a task is added. With an increase in the number of task requests in the SINs, the task execution time rises. When there are few tasks in the SINs, the task execution time of our algorithm is not much different from that of other algorithms. When the task increases gradually, our algorithm has more significant advantages than other algorithms. The main reason is that it is easier for the algorithm to identify relevant resources when there are few tasks, so the execution times of several algorithms are similar. The DQNRS algorithm...
algorithm can fit the Q-table through the neural network and has a better effect in a significant environmental state. The algorithm optimizes the task execution time so that DQNRS can identify the appropriate task and resource matching strategy after the number of tasks increases. Simultaneously, the DQNRS algorithm clusters resources in the resource preprocessing stage, so it can match resources faster and achieve a better execution time than other algorithms, such as DRL-VNE.

Figure 5 shows the change in the corresponding task execution success rate when the tasks of the SINs increase. Task execution failure is mainly attributed to the invisible waiting time and location coverage of multiple tasks after the tasks are allocated to resources. Among them, the DRL-VNE algorithm performs best, while the task execution success rate corresponding to the random algorithm decreases significantly with an increase in the number of tasks. Although the task execution success rate of the DQNRS algorithm also decreases, it can better maintain the task execution success rate because it considers the task execution time, resource usage, and coverage area as dynamic optimization objectives. Intelligent experience dynamically adjusts the optimization objectives according to the changes in the resource pool so that more tasks can be executed successfully.

As shown in Figure 6, the increase in tasks in SINs increases the demand for resources. The DQNRS algorithm proposed in this paper still has advantages compared with the ACO, random, FCFO, and DRL-VNE algorithms. The DQNRS algorithm performs variable granularity clustering of resources at the beginning of scheduling, aggregating resources into different types of resource clusters, which helps the scheduling algorithm quickly search for appropriate resources. Different resource granularities can provide different capabilities. The agent of the algorithm can match more appropriate resources for the tasks through continuous dynamic optimization. The algorithm improves the efficiency of resource use and reduces the amount of resource use.

As shown in Figure 7, the task execution time decreases with an increase in the number of satellite nodes in the SINs. With this increase, virtual resource clusters in the cloud environment continue to increase so that tasks have more choices and it is easier to identify appropriate resources that can reduce the execution time of tasks. The random algorithm and FCFO algorithm consider the overall situation of the tasks to a lesser extent, so the resource utilization efficiency is low. Although the task execution time can be reduced after the number of resources is increased, the effect
is not optimal. The DQNRS, DRL-VNE, and ACO algorithms can better optimize the task execution time with the expansion of resources. Both DQNRS and DRL-VNE can search the environment and perform better in a complex environment, but DQNRS can make better dynamic adjustments according to the change in environment, so it achieves better performance in a high dynamic space environment.

As shown in Figure 8, the completion rate of tasks increases with an increase in resources. When the number of tasks remains unchanged, the capacity and quantity provided by the cloud environment will increase accordingly after the increase in resources, so the success rate of tasks will increase. With fewer resources, the task completion rate of the DQNRS algorithm is only less than that of the DRL-VNE algorithm because the DQNRS makes dynamic adjustment decisions according to the optimization objectives. This process can continuously optimize the resource utilization rate. The more reasonable the resource application is, the higher the task completion rate is. In the process of increasing resources, although the completion rate is not as high as that of DRL-VNE, the DQNRS algorithm still performs well. The algorithm applies the neural network to adapt to the increasing environmental state. The algorithm resource processing stage can have a role in increasing the number of resources so that the search space can better match the search ability of the algorithm, improve the resource utilization, and increase the task completion rate.

Figure 9 illustrates the resource usage of different algorithms under resource growth. As seen in the figure, the increase in resources can reduce the use of resources. With the expansion of resources, more resources can be matched by tasks, and the decision-making process can better match the appropriate resources to reduce their use. The figure shows that the random algorithm randomly performs resource matching, and the resource utilization is not high, which makes the resource usage increase significantly compared with other algorithms. With the increase in resources, the DQNRS algorithm has advantages over other algorithms. The algorithm aggregates resources into resource clusters with different capabilities through variable granularity clustering of resources, effectively improving the search ability of resources and improving the probability of searching for more suitable resources. The multiobjective optimization of resource usage and location can effectively improve the resource utilization rate to reduce resource usage.
4. Conclusions

This paper investigates the resource scheduling problem in the SIN environment. We propose an SIN cloud computing architecture to share multisource and heterogeneous SIN resources, which provides a new idea for resource management. To reduce the complexity and diversity of cloud environment resources, we propose a variable granularity clustering algorithm, which reduces the resource search space in the cloud environment and improves resource utilization efficiency. We analyze the scheduling problem with regard to the task execution time, resource usage, and coverage location and propose an intelligent resource allocation algorithm, DQNRS, based on DQN technology. Based on the adaptive learning ability and decision-making characteristics of DQN in the environment, our algorithm can better adapt to SINs with dynamic changes and complex resources. We conducted several simulations to analyze the performance of the DQNRs. The experimental results show that although the completion rate of the proposed algorithm is weaker than that of the DRL-VNE algorithm, it performs better with regard to task execution time and resource usage. The performance is better than that of the ACO, random, and FCFO algorithms. In a future work, we plan to design a perfect network architecture and continue to design new scheduling algorithms to improve the resource utilization of SINs.

Data Availability

The data used to support the findings of this study are included within the article.

Conflicts of Interest

The authors declare that there is no conflict of interest regarding the publication of this paper.

Acknowledgments

This research was also partly supported by the National Natural Science Foundation of China under Grant 61671141.

References

[1] L. Wei, Y. Chen, M. Chen, and Y. Chen, “Deep reinforcement learning and parameter transfer based approach for the multi-objective agile earth observation satellite scheduling problem,” Applied Soft Computing, vol. 110, 2021.
[2] W. A. Haijiao, Y. A. Zhen, Z. H. Wugen, and L. I. Dalin, “Online scheduling of image satellites based on neural networks and deep reinforcement learning,” Chinese Journal of Aeronautics, vol. 32, no. 4, pp. 1011-1019, 2019.
[3] Y. He, G. Wu, Y. Chen, and W. A. Pedrycz, “A two-stage framework and reinforcement learning-based optimization algorithm for complex scheduling problems,” http://arxiv.org/abs/2103.05847.
[4] Y. Du, T. Wang, B. Xin, L. Wang, Y. Chen, and L. Xing, “A data-driven parallel scheduling approach for multiple agile earth observation satellites,” IEEE Transactions on Evolutionary Computation, vol. 24, no. 4, pp. 679–693, 2020.
[5] B. Yang, Y. Wu, X. Chu, and G. Song, “Seamless handover in software-defined satellite networking,” IEEE Communications Letters, vol. 20, no. 9, pp. 1768–1771, 2016.
[6] N. Zhang, S. Zhang, P. Yang, O. Alhussein, W. Zhuang, and X. S. Shen, “Software defined space-air-ground integrated vehicular networks: challenges and solutions,” IEEE Communications Magazine, vol. 55, no. 7, pp. 101–109, 2017.
[7] D. Zhu, H. Liu, T. Li et al., “Deep reinforcement learning-based task offloading in satellite-terrestrial edge computing networks,” in 2021 IEEE Wireless Communications and Networking Conference (WCNC), pp. 1–7, Nanjing, China, 2021.
[8] J. Wang, L. Zhao, J. Liu, and N. Kato, “Smart resource allocation for mobile edge computing: a deep reinforcement learning approach,” IEEE Transactions on Emerging Topics in Computing, vol. 9, no. 3, pp. 1529–1541, 2021.
[9] N. Bianchessi, J. F. Cordeau, J. Desrosiers, G. Laporte, and V. Raymond, “A heuristic for the multi-satellite, multi-orbit and multi-user management of Earth observation satellites,” European Journal of Operational Research, vol. 177, no. 2, pp. 750–762, 2007.
[10] Y. Liu, Y. W. Chen, and Y. J. Tan, “Modeling and algorithm for the new tasks’ arriving in multi-satellites dynamic scheduling,” System Engineering, vol. 25, no. 4, pp. 35–41, 2005.
[11] D. Habet, M. Vasquez, and Y. Vimont, “Bounding the optimum for the problem of scheduling the photographs of an agile earth observing satellite,” Computational Optimization and Applications, vol. 47, no. 2, pp. 307–333, 2010.
[12] M. Lemaître, G. Verfaillie, and N. Bataille, “Sharing the use of a satellite: an overview of methods,” in 5th international symposium on space mission operations and ground data systems, pp. 1–7, Tokyo, Japan, 1998.
[13] P. Wang, G. Reintelt, P. Gao, and Y. Tan, “A model, a heuristic and a decision support system to solve the scheduling problem of an earth observing satellite constellation,” Computers & Industrial Engineering, vol. 61, no. 2, pp. 322–335, 2011.
[14] B. Dilkina and B. Havens, Agile Satellite Scheduling via Perturbation Search with Constraint Propagation, Actenum Corporation, Vancouver Canada, 2005.
[15] Y. Li, M. Xu, and R. Wang, “Scheduling observations of agile satellites with combined genetic algorithm,” in Third International Conference on Natural Computation (ICNC 2007), pp. 29–33, Haikou, China, 2007.
[16] X. Geng, J. Li, W. Yang et al., “Agile satellite scheduling based on hybrid coding genetic algorithm,” in 2016 12th World Congress on Intelligent Control and Automation (WCICA), pp. 2727–2731, Guilin, China, 2016.
[17] F. Marinelli, S. Nocella, F. Rossi, and S. Smriglio, “A Lagrangian heuristic for satellite range scheduling with resource constraints,” Computers & Operations Research, vol. 38, no. 11, pp. 1572–1583, 2011.
[18] D. Eddy and M. Kochenderfer, “Markov decision processes for multi-objective satellite task planning,” in 2020 IEEE Aeronautics Conference on Intelligent Control and Automation (WCICA), pp. 1–12, Big Sky, MT, USA, 2020.
[19] P. Tangpattanakul, N. Jozefowiez, and P. Lopez, “A multi-objective local search heuristic for scheduling Earth observations taken by an agile satellite,” European Journal of Operational Research, vol. 245, no. 2, pp. 542–554, 2015.
[20] K. Sun, J. Li, Y. Chen, and R. He, “Multi-objective mission planning problem of agile Earth observing satellites,” in
16 Wireless Communications and Mobile Computing

Proceedings of the 12th International Conference on Space Operations, vol. 4, pp. 2802–2810, Stockholm, Sweden, 2012.

[21] L. Li, H. Chen, J. Li, N. Jing, and M. Emmerich, "Preference-based evolutionary many-objective optimization for agile satellite mission planning," IEEE Access, vol. 6, pp. 40963–40978, 2018.

[22] Z. Li and X. Li, "Current status and prospect of imaging satellite task dynamic scheduling methods," in 2016 8th International Conference on Intelligent Human-Machine Systems and Cybernetics (IHMSC), pp. 436–439, Hangzhou, China, 2016.

[23] X. Zhao, Z. Wang, and G. Zheng, "Two-phase neural combinatorial optimization with reinforcement learning for agile satellite scheduling," Journal of Aerospace Information Systems, vol. 17, no. 7, pp. 346–357, 2020.

[24] S. S. Mozer and G. B. Andrew, Reinforcement Learning: An Introduction, MIT Press, Massachusetts, 2005.

[25] T. P. Lillicrap, J. J. Hunt, A. Pritzel et al., "Continuous control with deep reinforcement learning," Computer Science, vol. 8, no. 6, p. A187, 2015.

[26] N. Liu, Z. Li, J. Xu et al., "A hierarchical framework of cloud resource allocation and power management using deep reinforcement learning," in 2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS), pp. 372–382, Atlanta, GA, USA, 2017.

[27] F. Zaman, S. Robitsch, Z. Wu, J. Keeney, S. van der Meer, and G. M. Muntean, "A heuristic correlation algorithm for data reduction through noise detection in stream-based communication management systems," in 2014 IEEE Network Operations and Management Symposium (NOMS), pp. 1–8, Krakow, Poland, 2014.

[28] X. Liu, J. Yu, Z. Feng, and Y. Gao, "Multi-agent reinforcement learning for resource allocation in IoT networks with edge computing," China Communications, vol. 17, no. 9, pp. 220–236, 2020.

[29] S. Vimal, M. Khari, N. Dey, R. G. Crespo, and Y. Harold Robinson, "Enhanced resource allocation in mobile edge computing using reinforcement learning based MOACO algorithm for HOT," Computer Communications, vol. 151, pp. 355–364, 2020.

[30] M. Li, F. R. Yu, P. Si, W. Wu, and Y. Zhang, "Resource optimization for delay-tolerant data in blockchain-enabled IoT with edge computing: a deep reinforcement learning approach," IEEE Internet of Things Journal, vol. 7, no. 10, pp. 9399–9412, 2020.

[31] G. Rjoub, J. Bentahar, O. Abdel Wahab, and A. Saleh Bataineh, "Deep and reinforcement learning for automated task scheduling in large-scale cloud computing systems," Concurrency and Computation: Practice and Experience, vol. 33, 2020.

[32] S. Arisdakessian, O. A. Wahab, A. Mourad, H. Otrok, and N. Kara, "FoGMatch: an intelligent multi-criteria IoT-Fog scheduling approach using game theory," IEEE/ACM Transactions on Networking, vol. 28, no. 4, pp. 1779–1789, 2020.

[33] M. Cheng, J. Li, and S. Nazarian, "DRL-cloud: deep reinforcement learning-based resource provisioning and task scheduling for cloud service providers," in 2018 23rd Asia and South Pacific Design Automation Conference (ASP-DAC), pp. 129–134, Jeju, Korea (South), 2018.

[34] C. Qiu, H. Yao, F. R. Yu, F. Xu, and C. Zhao, "Deep q-learning aided networking, caching, and computing resources allocation in software-defined satellite-terrestrial networks," IEEE Transactions on Vehicular Technology, vol. 68, no. 6, pp. 5871–5883, 2019.

[35] B. Deng, C. Jiang, H. Yao, S. Guo, and S. Zhao, "The next generation heterogeneous satellite communication networks: Integration of resource management and deep reinforcement learning," IEEE Wireless Communications, vol. 27, no. 2, pp. 105–111, 2020.

[36] X. Liao, X. Hu, Z. Liu et al., "Distributed Intelligence: a verification for multi-agent DRL-based multibeam satellite resource allocation," IEEE Communications Letters, vol. 24, no. 12, pp. 2785–2789, 2020.

[37] X. Hu, S. Liu, R. Chen, W. Wang, and C. Wang, "A deep reinforcement learning-based framework for dynamic resource allocation in multibeam satellite systems," IEEE Communications Letters, vol. 22, no. 8, pp. 1612–1615, 2018.

[38] N. Garg, M. Sellathurai, and T. Ratnarajah, "In-network caching for hybrid satellite-terrestrial networks using deep reinforcement learning," in ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 8797–8801, Barcelona, Spain, 2020.

[39] C. Jiang and X. Zhu, "Reinforcement learning based capacity management in multi-layer satellite networks," IEEE Transactions on Wireless Communications, vol. 19, no. 7, pp. 4685–4699, 2020.

[40] L. Dalin, W. Hajiioo, Y. Zhen, G. Yang, and S. Shi, "An online distributed satellite cooperative observation scheduling algorithm based on multiagent deep reinforcement learning," IEEE Geoscience and Remote Sensing Letters, vol. 18, no. 11, pp. 1901–1905, 2021.

[41] N. Cesa-Bianchi, C. Gentile, G. Lugosi, and G. Neu, "Boltzmann exploration done right," http://arxiv.org/abs/1705.10257.

[42] P. Zhang, C. Wang, N. Kumar, and L. Liu, "Space-air-ground integrated multi-domain network resource orchestration based on virtual network architecture: a DRL method," IEEE Transactions on Intelligent Transportation Systems, 2021.

[43] C. Mateos, E. Pacini, and C. G. Garino, "An ACO-inspired algorithm for minimizing weighted flowtime in cloud-based parameter sweep experiments," Advances in Engineering Software, vol. 56, pp. 38–50, 2013.

[44] A. Ponraj, "Optimistic virtual machine placement in cloud data centers using queuing approach," Future Generation Computer Systems, vol. 93, no. 3, pp. 338–344, 2019.