A Generic Markov Decision Process Model and Reinforcement Learning Method for Scheduling Agile Earth Observation Satellites

Yongming He, Lining Xing, Yingwu Chen, Witold Pedrycz, *Life Fellow, IEEE*, Ling Wang, and Guohua Wu

***Abstract*—We investigate a general solution based on reinforcement learning for the agile satellite scheduling problem. The core idea of this method is to determine a value function for evaluating the long-term benefit under a certain state by training from experiences, and then apply this value function to guide decisions in unknown situations. First, the process of agile satellite scheduling is modeled as a finite Markov decision process with continuous state space and discrete action space. Two subproblems of the agile Earth observation satellite scheduling problem, i.e., the sequencing problem and the timing problem are solved by the part of the agent and the environment in the model, respectively. A satisfactory solution of the timing problem can be quickly produced by a constructive heuristic algorithm. The objective function of this problem is to maximize the total reward of the entire scheduling process. Based on the above design, we demonstrate that Q-network has advantages in fitting the long-term benefit of such problems. After that, we train the Q-network by Q-learning. The experimental results show that the trained Q-network performs efficiently to cope with unknown data, and can generate high total profit in a short time. The method has good scalability and can be applied to different types of satellite scheduling problems by customizing only the constraints checking process and reward signals.**

***Index Terms*—Agile Earth observation satellite (AEOS), Markov decision process (MDP), Q-learning, reinforcement learning (RL), task scheduling.**

Manuscript received May 26, 2020; accepted August 15, 2020. Date of publication September 18, 2020; date of current version February 17, 2022. This work was supported in part by the National Natural Science Foundation of China under Grant 71701203 and Grant 61873328; and in part by the State Key Laboratory of Digital Manufacturing Equipment and Technology under Grant DMETKF2020030. The work of Yongming He was supported by the China Scholarship Council. This article was recommended by Associate Editor X. You. *(Corresponding author: Guohua Wu.)*

Yongming He, Lining Xing, and Yingwu Chen are with the College of Systems Engineering, National University of Defense Technology, Changsha 410073, China (e-mail: heyongming10@hotmail.com; xinglining@gmail.com; ywchen@nudt.edu.cn).

Witold Pedrycz is with the Department of Electrical and Computer Engineering, University of Alberta, Edmonton, AB T6G 2V4, Canada, and also with the Systems Research Institute, Polish Academy of Sciences, 01447 Warsaw, Poland (e-mail: wpedrycz@ualberta.ca).

Ling Wang is with the Department of Automation, Tsinghua University, Beijing 100084, China (e-mail: wangling@mail.tsinghua.edu.cn).

Guohua Wu is with the School of Traffic and Transportation Engineering, Central South University, Changsha 410075, China (e-mail: guohuawu@csu.edu.cn).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TSMC.2020.3020732.

Digital Object Identifier 10.1109/TSMC.2020.3020732

I. Introduction

OWING to the fact that aerospace science and technology is developing at full speed, satellites play an increasingly significant role in many areas of our society. In particular, the development of imaging payload technology has brought the application of the agile Earth observation satellites (AEOSs) to a new era. For example, AEOSs can capture pictures of higher resolution, so they can be applied to wide areas that require accurate image information, such as intelligent transportation, emergency response, and military operation. Users' demands are rising sharply while the constraints of scheduling satellites become complex, which brings challenges in the AEOS scheduling.

In terms of the current level of AEOS management, scheduling processes for most of AEOSs are run in operation centers. The operation center collects all users' demands and then decides which demands should be satisfied and when they are fulfilled considering all constraints of the AEOS scheduling problem. Then the operation center generates a list of command codes according to the scheduling result and sends them to the corresponding AEOS. The AEOS executes these codes and collects images accordingly. Finally, these images are downloaded and processed into satellite application products for users [1].

As a core component in an AEOS operation system, an effective scheduling algorithm is pivotal to improve the efficiency of the whole system. However, the AEOS scheduling problem has been proved to be NP-hard [2]. On the other hand, different design on AEOS hardware brings varying constraints to the scheduling process, which leads to the fact that one needs to design a customized algorithm for each type of satellite. Nowadays, the capabilities of AEOSs have been greatly developed, and the optimization model of the AEOS scheduling problem becomes more complicated. It could be time-consuming and inefficient to simplify the model and design customized rules for scheduling AEOSs. In view of this, proposing a novel and generic method for scheduling AEOSs without the manual design of the rules in the algorithm is meaningful.

As we all know, machine learning is a methodology that uses experience or data to improve the effectiveness of algorithms in complex applications. It achieves fast and accurate decision-making in various situations by training from experience or data [3]. Supervised learning, unsupervised learning, and reinforcement learning (RL) are the main paradigms of machine learning. Supervised learning could support decision-making, e.g., AlphaGo summarizes the winning probability of choosing different actions in any state by learning labeled human games [4]. However, this approach requires quantities of labeled data, and the results are strongly coupled with the characteristics of the data set [5], [6]. Unsupervised learning is widely used in clustering, pattern recognition, and feature extraction, but not adept in decision-making. RL is a methodology for learning the policy from the state-action pairs and the feedback from the environment. Therefore, no labeled data or model needed when training RL.

Problems that can be solved by RL must be described as a finite Markov decision process (MDP), and the rewards of all actions in different states must be clearly explained. The following prerequisites ensure that the RL approaches fit the AEOS scheduling problem well: 1) scheduling constraints and AEOS capabilities do not change once an AEOS has been manufactured [1]; 2) the AEOS scheduling problem can be formulated as a sequential decision-making problem [7]–[9]; and 3) the objective function is explicit. For any set of values of decision variables, a unique objective function value can be obtained.

The AEOS scheduling problem is a decision-making problem of discrete variables, which is suitable to be solved by supervised learning or RL algorithms. However, it is difficult to collect enough labeled data for the AEOS scheduling problem. Hence, modeling the AEOS scheduling problem as an MDP and then solving it by an RL algorithm is a potential way to deal with it.

In this study, we focus on scheduling AEOSs by introducing an effective RL method. An MDP is established to describe the AEOS scheduling problem. Agent in the MDP learns from a few sets of data and make decisions with a short time. The experiments show that RL produces better results than other advanced methods.

The major contributions of this article are summarized as follows.

1) A general MDP for AEOS scheduling problems is constructed. We put forward a novel framework of the MDP, which splits the AEOS scheduling problem to a sequencing problem and a timing problem. These two subproblems are processed in different parts of the MDP. The elements of the MDP are designed according to the common points of different AEOS scheduling problems, which guarantees the MDP describes this type of problems well.
2) A model-free offline RL algorithm is built for learning the estimated long-term value to guide the decision-making process. In the training process, a pruning strategy based on the characteristics of the scheduling problem and a rule for avoiding infinite loops in each cycle are designed for improving efficiency.
3) A generic framework that combines deep Q-learning and an advanced heuristic algorithm is built for scheduling AEOSs. These two algorithms solve the sequencing problem and the timing problem in the AEOS scheduling problem, respectively.
4) Extensive comparisons with other algorithms verified the feasibility and effectiveness of the proposed algorithm. Experimental results show that the proposed method has higher convergence speed and accuracy than other advanced RL algorithms. Meanwhile, the proposed algorithm surpasses the other two advanced algorithms in the AEOS scheduling field.

The remainder of this article is structured as follows. Section II reviews and analyzes existing studies on the AEOS scheduling problem as well as RL approaches. Section III builds a finite MDP to formulate the AEOS scheduling problem. Section IV introduces a framework and a model-free method for solving this problem. Section V reports on experiments completed under various conditions, and then compares the proposed method with other approaches.

II. Literature Review

"Soft management" is popular when the topic of scheduling satellites appears. Barry developed an expert system for scheduling and controlling Rockwell Satellite [10], [11], to reduce costs and decrease manual intervention. However, it is time-consuming and laborious to complete the system and maintain it.

In 2000, Wolfe and Sorensen [12] proposed the window-constrained packing problem model, where for the first time an expert has used an optimization model to describe satellite scheduling problems. Since then, AEOS scheduling problems are considered as an optimization problem, and the algorithms applied to these problems can be divided into three categories: 1) mathematical programming algorithms; 2) heuristic algorithms; and 3) meta-heuristic algorithms. Mathematical programming, such as tree search [7], [13], branch-and-bound [14], [15], branch-and-price [16], and dynamic programming [2], [17] can only get satisfactory solutions for small-scale scenarios because the time complexity of these algorithms is usually high. Because of the low computational complexity of heuristic approaches and limited computing power and time, heuristic approaches are widely applied to practical problems, such as EO-1 in the USA [18]–[20], Pleiades in France [7], [21], [22], and FireBIRD in Germany [8], [23]. He *et al.* [24] developed a heuristic algorithm based on the density of residual tasks (HADRTs) and proved the optimality of this algorithm under certain assumptions. However, it is hard to find a suitable heuristic for a specific problem, or the efficiency of this type of algorithm is difficult to verify. The idea of using meta-heuristics to solve decision problems originated in the 1950s [25], while it has developed rapidly in the past decade. Adaptive large neighborhood search meta-heuristic (ALNS) [26], [27], genetic algorithms [28], [29], ant colonies, and their variants [30] are considered for scheduling AEOSs in the previous studies. In some scenarios, these algorithms achieve better results than some mathematical algorithms and heuristic algorithms, but two drawbacks imply they are not really popular in practical problems: 1) it is not easy to find common parameters and functions that can produce good results for different input data [31] and 2) in some complex scenarios, this type of algorithm is difficult to converge in an acceptable time. Overall, no matter which of the above three types of methods are used, their decision rules are artificially formulated and will not change during the calculation process.

In recent years, RL has made breakthroughs in solving sequential decision-making problems, such as Texas Hold'em poker game [32], autonomous underwater vehicle control [33], unmanned aerial vehicle control [34], while the most well-known one is AlphaGO [4]–[6]. In the scheduling field, Nazari *et al.* [35] developed the pointer network as the actor-network of the actor–critic algorithm to solve the vehicle routing problem. Furthermore, we have identified some interesting studies in applying RL to satellite scheduling: Usaha and Barria [36] compared an actor–critic algorithm with a value-based algorithm in the resource allocation problem of LEO satellite systems. They designed an explicit function to describe the value function. Haijiao *et al.* [37] viewed the online scheduling problem of image satellites as a dynamic stochastic knapsack problem and then solved this problem by asynchronous advantage actor–critic. Issues in these works can be easily described as MDPs and are solved by designing novel RL algorithms. However, some complex practical problems are hard to be dealt with this way, because it is hard to build suitable MDPs for these problems which can be handled efficiently by RL algorithms.

III. Finite Markov Decision Process for the AEOS Scheduling Problem

A. Problem Description

The AEOS operation center receives imaging requests from various users. After pretreatment, each request is converted into a set of data for calculation and decision. This data set is called a task, which contains attributes about visible time windows, imaging duration, profit, and others. According to the objective, constraints, and capabilities of the problem, the system exports a scheme for satellite activities based on a scheduling algorithm. Finally, the command codes are sent to the satellite, and the satellite executes tasks according to the command codes. The data flow in the operation center is shown in Fig. 1.

Thus, the AEOS scheduling problem can be illustrated as follows. Given a set of attributes for each task, and the capabilities of the AEOS, a scheme is needed to guide which and when the tasks should be executed by the AEOS. The goal of this process is to maximize the value of the objective function while meeting all constraints.

General objective function and constraints in AEOS scheduling problems are widely discussed in [14], [24], and [26]. In summary, the objective function considered in this work is the total profit of all scheduled tasks, while constraints are listed as follows.

1) Each task is executed at most once.
2) Any task can only be executed within its visible time windows.

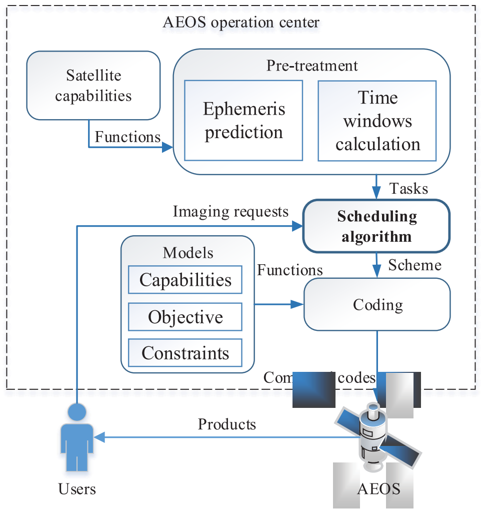


Fig. 1. Data flow of the AEOS operation center.

3) The execution time windows of any two tasks are not overlapping.
4) The time interval between the execution windows of any two consecutive tasks are not shorter than the transition time required.
5) The consumed memory storage by the tasks in the scheduling scheme is not greater than the storage capacity of the satellite.

Considering the current capabilities of relay satellites and data transmission stations [38], as well as the limitations of the data transmission process mentioned in [39] and [40], it is possible for an AEOS returning all collected imaging data to the ground center in each scheduling period. Furthermore, other state constraints related to external conditions, such as light conditions and constraints related to satellite hardware capabilities have been treated in the pretreatment, for reducing the uncertainty and shrinking the solution space [41], [42].

Capabilities of AEOS in this research are summarized as follows [14], [24], [26]. The parameters of these capabilities are specified in Section V-A1.

1) The pointing angle of each task is restricted in a range.
2) The battery is sufficient because the AEOS can charge all the time.
3) The memory storage capacity of the AEOS is limited.
4) The maneuvering ability of the satellite is limited, which relates to the fourth constraint in the list of constraints.

B. Modeling Process

The problem can be decomposed into two phases: 1) the sequencing problem and 2) the timing problem [12].

Once the task sequence has been determined, there are several algorithms for the timing problem. Dynamic programming [17] and some constructive heuristic algorithms [7], [18], [23] perform well in determining the execution time of each task of AEOS with a certain task sequence. In our framework, we choose a novel heuristic algorithm to deal with the second phase of the AEOS scheduling problem, which is HADRT in [24].

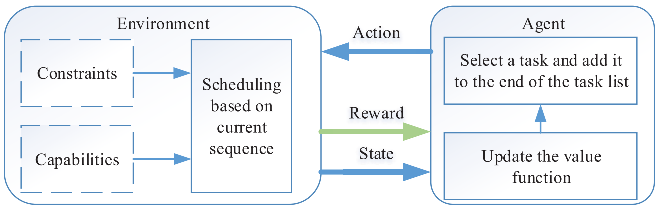

Environment
Constraints
Capabilities
Scheduling based on current sequence
Action
Reward
State
Agent
Select a task and add it to the end of the task list
Update the value function

Fig. 2. Agent-environment interaction of the AEOS scheduling problem.

The sequencing problem is the problem that arises after the satellite owns agility, which also leads the AEOS scheduling problem to an NP-hard problem. In this study, the sequencing problem is built as a time series decision problem, and then design an efficient approach to deal with it.

The schematic of the MDP for the AEOS scheduling problem is shown in Fig. 2. The state, action, reward, and value function of the MDP are introduced in this section.

In the beginning, the system is in the initial state S_0. The agent performs an action A_0 according to S_0. After the environment receives A_0, the reward R_1 of action A_0 is calculated, and the state S_1 (the state of next phase) is refreshed. The agent continuously outputs actions A_i based on S_i until the terminal state has been reached. Each interaction between the agent and the environment is called a time step.

1) State: State space $\boldsymbol{S}$ is the set of state. It can be formed as follows:

$$\boldsymbol{S} = \{S_0, S_1, \ldots, S_i, \ldots\} \tag{1}$$

S_i is the state of the ith time step. Each state contains a set of attributes that describe the situation of the AEOS scheduling problem at a certain time step. The details of the state are shown as follows:

$$S_i = \left\{x_i^t = \left(g^t, p^t, v_i^t, l_i^t\right) | t = 0, 1, \ldots, N\right\} \tag{2}$$

where g^t is the geographic information of task t. It records the longitude and latitude of the request, expressed as follows:

$$g^t = \left(\text{lon}^t, \text{lat}^t\right) \tag{3}$$

p^t is the profit of task t after executing it. g^t and p^t provide by users, and they do not change over time. v_i^t is the number of remaining visible time windows of task t at the ith time step. After pretreatment, we consider tasks without available visible time window as invalid tasks. So each task in the task list has one or more visible time windows during the scheduling period. Tasks can not be executed outside their visible time windows, so v_i^t is an important index when making decisions at a certain time step i. l_i^t is the label indicating whether task t is selected before time step i

$$l_i^t = \begin{cases} 0, & \text{task } t \text{ is selected before time step } i \\ 1, & \text{task } t \text{ is unselected before time step } i. \end{cases} \tag{4}$$

The state at any time step S_i is described according to the above design.

It is also necessary to define a rule to judge whether a state is a terminal state. According to the characteristics of the AEOS scheduling problem, if all tasks have no chance to be scheduled after one time step, then the state at this time step is the termination state. It could be judged by the following equation:

$$\sum_{t=0}^{N-1} v_i^t l_i^t = 0. \tag{5}$$

2) Reward: The reward which the agent receives from the environment is not always equal to the profit of the selected action. The reward in this model is the increment of the total profit after taking an action

$$\boldsymbol{R} = \{R_1, R_2, \ldots, R_i, \ldots | R_i = r\} \tag{6}$$
$$r = f(\boldsymbol{X}_i) - f(\boldsymbol{X}_{i-1}). \tag{7}$$

$\boldsymbol{X}_i$ is the matrix of decision variables at time step i. $f(\boldsymbol{X}_i)$ indicates the total profit at time step i, which could be calculated by an advanced algorithm for the timing problem. The optimality of HADRT under certain assumptions has been proved in reference [24]. The core idea of this algorithm is selecting the tasks by the heuristic function, which is calculated by the remaining time length and average working time of each task.

According to our design, at each time step the environment receives an action, it will export the corresponding reward and state automatically.

3) Action: Action in the model is "select a task" or "continue." The action select a task is to choose a task for the timing problem, and the environment will add it to the tail of the queue of selected tasks for scheduling. "continue" is designed for avoiding local-optimal. Fig. 3 illustrates an example to support this point.

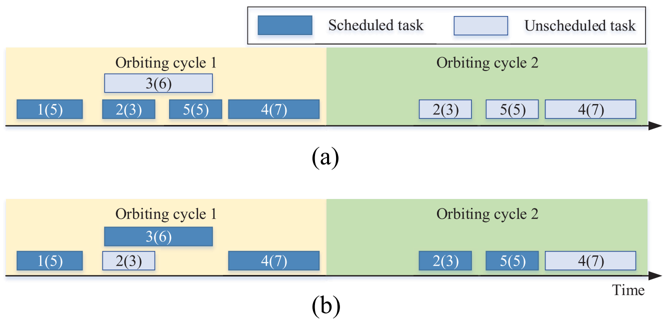


Fig. 3. Example for explaining the difference between two settings in actions. (a) No action in the action space apart from select a task. (b) Insert continue action in the action space.

As shown in Fig. 3, The orbiting cycles are due to the periodicity of the satellite orbit. Each orbiting cycle is an independent period for scheduling. The number in the parentheses indicates the profit after executing this task, while the corresponding task ID is shown outside the parentheses. In Fig. 3(a), the total profit of these two orbiting cycles is less than that of Fig. 3(b). This example tells that: without continue in the action space, the overall result will fall into a local optimum. Thus, a forward-looking agent is needed for solving the AEOS scheduling problem.

In summary, the action space of this problem is discrete, and the size of the action space $|\boldsymbol{A}|$ is the number of input

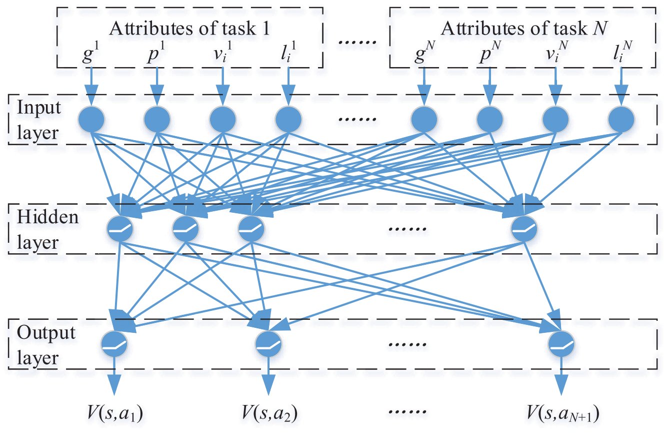


Fig. 4. Structure of the value function.

tasks N plus one

$$|\boldsymbol{A}| = N + 1. \tag{8}$$

4) Value Function: Most RL algorithms contain a value function [3]. It is commonly a function of a state-action pair that estimates how good the long-term benefit is after selecting an action in a state. Usually, the value function defines the average return of the action in the long term. According to the Bellman optimality formula, the value function can be learned through the following:

$$q(s, a) = E\Big(R_{i+1} + \gamma \max_{a'} q\big(s_{i+1}, a'\big)|S_i = s, A_i = a\Big). \tag{9}$$

In our problem, it is clear that the state space is continuous, while the action space is discrete. Therefore, tabular methods are not suitable for this problem. This article builds the value function by the form of a fully connected back propagation neural network.

The structure of the value function is put forward in Fig. 4. This neural network contains an input layer, a hidden layer, and an output layer. The input of this network is the state of time step i, containing attributes of all tasks. The output is a vector that provides the value for each action at the state s. The gap between the output value of the network and the real value is getting smaller through iterating by RL.

C. Discussion About the Proposed MDP

So far, the MDP with a heuristic algorithm is built. The proposed MDP extends the connotation of the traditional MDP.

First, the MDP treats the objective function and constraints of the AEOS scheduling problem as a black box. The AEOS scheduling problem is divided into two subproblems to give full play to the advantages of heuristic algorithms and RL algorithms.

Second, states and actions in the proposed MDP are designed carefully according to the characteristics of the AEOS scheduling problem, which supports the proposed algorithm to find the solution quickly and accurately.

Third, an approach to calculate the reward is created, using the results obtained by the advanced AEOS scheduling algorithm. Compared with the approaches for designing reward function which widely used methods in literatures, such as artificially designed labels or domain knowledge-inspired functions [34], [43], this method is more versatile and precise in complex real-world problems.

Finally, it is difficult to find an explicit function to indicate the long-term rewards of the AEOS scheduling problem. A neural network is used in this article to represent the value function, and the input and the output of the neural network are designed to make it suitable for scheduling problems.

IV. Deep Q-Learning for the Problem

The approach for solving the MDP in this work is deep Q-learning, where the Q-network is used to represent the value function. However, directly applying the traditional deep Q-learning from other literatures cannot efficiently handle the proposed MDP and the AEOS scheduling problem. The proposed algorithm has the following two improvements over traditional deep Q-learning.

1) An efficient pruning strategy is designed during the process of choosing an action. A strategy of wiping out unavailable and unrewarded actions according to the characteristics of the problem is applied to improve the efficiency in the action selection of both training and testing process.
2) A novel framework for multiple scenarios is proposed. Unlike the classic problems that have been solved by deep Q-learning perfectly, the initial states of the AEOS scheduling problem vary in different scenarios. We embed the procedure of randomly generating scenes into the framework of the training process of the deep Q-learning, which allows the Q-network to be generalized to as many unknown scenes as possible.

A. Training Process

Unlike the RL process for the game of Go or Atari, the initial states are different when scheduling AEOSs, because the requests from users vary in different scheduling periods. The state space is continuous in our case, so it is impossible to traverse all states during the training process.

Algorithm 1 shows the training process of the RL agent. The Q-network is updated once an action is taken and the corresponding reward is obtained. In each scenario, the agent will run a certain number of episodes and update the Q-network continuously. The agent repeats this process in different scenarios, and finally gets the value function for decisions in unknown situations.

When training the Q-network with the above algorithm, *index* is randomly generated at first. As shown in line 10 in Algorithm 1, by comparing the random number *index* with the threshold value ε, different policies in choosing an action are taken.

1) Pruning Strategy for Choosing Actions: In theory, the action space of each time step is $N + 1$. However, there are two situations in which task t is unavailable at time step i.

1) *Visibility Constraint:* There is no visible time window for task t at time step i, i.e., $v_i^t = 0$.

Algorithm 1: Training Process

```
1  foreach training scenario do
2      Import a new scenario
3      foreach episode do
4          Parameters initialization
5          repeat
6              $index \leftarrow$ random()
7              Set the threshold $\varepsilon$
8              $Q \leftarrow$predict($s$)
9              Get the available task list $AL$ under state $s$
10             $a \leftarrow \begin{cases} \text{Randomly taken from } AL & index \le \varepsilon \\ \arg\max Q(A)|A \in AL & \text{others} \end{cases}$
11             Refresh and record state $s'$
12             Calculate reward $r$
13             Train Q-network with experiment replay
14         until s is the terminal state or the program reashes the maximum number of iterations;
```

Algorithm 2: Train the Q-Network With Experiment Replay

```
   Input: The set of state $s$, state $s'$, action $a$, and reward
          $r$;  the initial Q-network
   Output: The updated Q-network
1  Add $s, s', a, r$ to the experience set
2  Initial parameters
3  for i = 1 : min(memory length, size of batch) do
4      load a record from the experience set
5      add the state $s$ to the set input_data
6      $Q(s, a) \leftarrow \max(\text{predict}(s'))$
7      if $s'$ is the terminal state then
8          $targets \leftarrow r$;
9      else
10         $targets \leftarrow r + \gamma * Q(s, a)$;
11     Update the gradient of the Q-network based on
       input_data and targets.
```

2) *Uniqueness Constraint:* Task t has been arranged at a previous time step, i.e., $l_i^t = 0$.

The action space could be pruned according to the above constraints. Each time the agent only selects the available tasks for reducing the search space of the sequencing problem. At the same time, the complexity of the constraint check in the timing problem can be reduced.

The uniqueness constraint and the visibility constraint are common in AEOS scheduling problems, so this pruning strategy has generality in various AEOS scheduling applications.

2) Training the Q-Network With Experiment Replay: The proposed algorithm constructs labels for the neural network through constant training. The experience replay mechanism [44], [45] is used to store historical training data and fit the network in a solid way. The pseudo-code of training the Q-network with experiment replay is given as Algorithm 2.

Algorithm 3: Testing Process

```
1  foreach testing scenario do
2      Import a new scenario
3      Parameters initialization
4      repeat
5          $Q \leftarrow$ predict($s$)
6          the available task list $AL$ under state $s$
7          $a \leftarrow \arg\max Q(A)|A \in AL$
8          Refresh and record state $s'$
9      until s is the terminal state;
10     Calculate and output the total profit
```

We save a list of tuple $\{s, s', a, r\}$ during the training process as the training data. The parameters of the Q-network are constantly updated during the training process. Lines 5–10 implement (9), and line 11 updates the parameters in the Q-network with the gradient descent method.

B. Testing Process

The differences between the training process and the testing process are summarized as follows.

1) No iteration needed in the same scene during the testing process.
2) No update needed for the Q-network during the testing process, so it is no longer necessary to record the experience history after training.
3) Different strategies for choosing tasks. During the testing process, the agent always chooses the action with the highest Q value. During the training process, this step contains two possibilities: a) "exploitation" to select the action with the highest value or b) exploration to select an action randomly.

The pseudo-code of the testing process is shown in Algorithm 3.

C. Complexity Analysis

1) Time Complexity: Calculations in the agent include choosing actions and training the network. It can be easily seen from Algorithm 1 that the time complexity of selecting an action is $O(N)$, where N is the number of input tasks. The time complexity of training once for a batch with batch size b is

$$\mathrm{TC}_{NN} = b(|\boldsymbol{S}|n_{\text{hid}} + n_{\text{hid}}|\boldsymbol{A}|) = O(N). \tag{10}$$

Calculations in the environment include updating status and feeding back the reward. The time complexity of these calculations is equivalent to that of HADRT, which is $O(N^2)$. The time complexity of the training process under the conditions of c training sets and e epochs is

$$\mathrm{TC}_{\text{train}} = ceN(\mathrm{TC}_{\text{act}} + m\mathrm{TC}_{NN} + \mathrm{TC}_{env}) = O\Big(N^3\Big). \tag{11}$$

Since the testing process only needs to call the result of the Q-network every time the action is selected, the time complexity of the testing process is

$$\mathrm{TC}_{\text{test}} = N\left(\mathrm{TC}_{\text{act}} + \frac{1}{b}\mathrm{TC}_{NN} + \mathrm{TC}_{env}\right) = O\left(N^3\right). \quad (12)$$

The training time and the testing time of the algorithm both increase polynomially as the task size N increases. Although constants, such as c, e, m, b, and n_{hid} also directly affect the calculation time of the algorithm, this does not hinder the scalability of the algorithm in large-scale scenarios.

2) Space Complexity: During the training process, the algorithm maintains an experience set and a network. By contrast, only the network needs to be maintained during the testing process. The role of the experience set is to store training history data, which contains m records, and each of the records contains the information of state s, state s', action a, and reward r of a time step

$$\mathrm{SC}_{\text{exp}} = m\left|\left\{s, s', a, r\right\}\right| = 2m(5N + 1) = O(N). \quad (13)$$

According to the design of Section III, the size of the state space and the action space is the number of nodes in the input and output layer of the network, respectively. Assuming that the number of nodes in the hidden layer is n_{hid}, and each node contains two parameters, the spatial complexity of the neural network is

$$\mathrm{SC}_{NN} = 2(|\boldsymbol{S}| + n_{\text{hid}} + |\boldsymbol{A}|) = O(N). \quad (14)$$

Hence, the space complexity of the training process is

$$\mathrm{SC}_{\text{train}} = \mathrm{SC}_{\text{exp}} + \mathrm{SC}_{NN} = O(N). \quad (15)$$

The space complexity of the testing process is

$$\mathrm{SC}_{\text{test}} = \mathrm{SC}_{NN} = 2(n_{\text{hid}} + 6N + 1) = O(N). \quad (16)$$

D. Discuss About the Algorithm

In this section, an improved deep Q-learning is applied to solve the AEOS scheduling problem. The objective of the AEOS scheduling problem is to maximize the total profit of the scheduled tasks, while that of the training Q-network is to minimize the mean square error between the real value and the predicted value under a certain state-action pair. Deep Q-learning is a value-based off-line RL method, and the policy in the training process is deterministic. So long as this value function is fitted well, the optimality of the proposed algorithm could be achieved according to Bellman equation [3], [46].

Unfortunately, we cannot guarantee the value function reaches its optimal value under the limited iterations of training, because the AEOS scheduling problem is an NP-hard problem, and the optimality of HADRT in the environment of the MDP cannot be guaranteed under all circumstances. Thus, we cannot theoretically prove that the algorithm converges to the optimal solution, but the experiments show that results obtained by this approach are fairly competitive.

V. Computational Experiments

Here, we apply the proposed algorithm to several scenarios of the AEOS scheduling problem. Several algorithms are used to compare with our method to verify the effectiveness of the proposed method.

TABLE I
Orbital Elements of the Satellite

$a\,(km)$	$e\,(°)$	$i\,(°)$	RAAN (°)	$\omega\,(°)$	$m\,(°)$
7200	0	96.576	175	0	0

A. Design of Scenarios

1) Design of the AEOS: For the stability of image quality, the sun-synchronous orbit is popular for AEOSs, with the orbital height between 600 and 1000 km. The orbital elements of the AEOS in the experiments are shown in Table I.

The scheduling horizon is 24 h. Common constraints and capabilities in this type of problem are considered, which are well defined in Section III. The parameters of the AEOS capabilities used in the experiments are summarized as follows.

1) The range of pitch angle and roll angle are [−45°, 45°].
2) The maximum storage is 750TB.
3) The details of the transition time function are described in [47].

2) Design of Tasks: We designed two types of distribution in tasks: 1) regional and 2) global.

Tasks in the "regional distribution" are located randomly in a region of 3° N–53° N, 73° E–133° E. Tasks in the "Global distribution" are located randomly in a region of 65° S–65° N, 180° W–180° E. Scenarios in regional distribution are able to test the efficiency of algorithms when the distribution of tasks is centralized, while scenarios in global distribution are able to test the performance of algorithms when the task is evenly distributed around the world. We tested the scale of tasks from 100 to 400 in regional distribution and from 100 to 600 in global distribution for comparing the proposed algorithm with other algorithms. The visible time windows of each task are calculated by the function of the position of the task and the ephemeris of the satellite, in the pretreatment process.

In addition to the geographical location of each task, imaging duration, and profit are also necessary for describing a task. For most AEOSs, the imaging duration is proportional to the area covered by the user's request. Assume that all users' requests are point targets, which means the imaging duration is a fixed number only related to the hardware design of the satellite. We set the imaging duration of each task as 5 s in the experiments.

The profit of each task is a number given by the user who submitted the corresponding request to the AEOS operation center. In order to standardize the description of the profit, it is usually discretized into ten levels to describe the importance of a task, taking an integer ranging from 1 to 10. In the experiments, we assign the value of the profit for each task randomly within this range.

3) Design of Parameters of the Algorithm: In the proposed algorithm, the Q-network is designed to evaluate the long-term benefit of taking an action in any state. The parameters of the Q-network are related to the task size in the scenario, see Table II.

Other parameters in the algorithm are summarized as follows.

TABLE II
PARAMETERS IN THE Q-NETWORK

Scenarios	input layer	hidden layer	output layer	Sum
Regional_100	50100	10100	10201	70401
Regional_200	100100	10100	20301	130501
Regional_300	150100	10100	30401	190601
Regional_400	200100	10100	40501	250701
Global_100	50100	10100	10201	70401
Global_200	100100	10100	20301	130501
Global_300	150100	10100	30401	190601
Global_400	200100	10100	40501	250701
Global_500	250100	10100	50601	310801
Global_600	300100	10100	60701	370901

1) Number of neurons in the hidden layer of the Q-network: 100.
2) The discount rate γ in the process of training the Q-network: 0.9.
3) The exploration rate ε: 0.2.
4) Maximum number of experiences algorithm has stored m: 10 000.
5) Number of experiences for training per batch b: 100.
6) Number of iterations in each scenes e: 20.
7) Number of training scenes c: 20.

The experiments were coded by Python. All experiments is implemented and compared on a laptop with Intel Core i7-8750H CPU @ 2.20 GHz, 16 GB RAM.

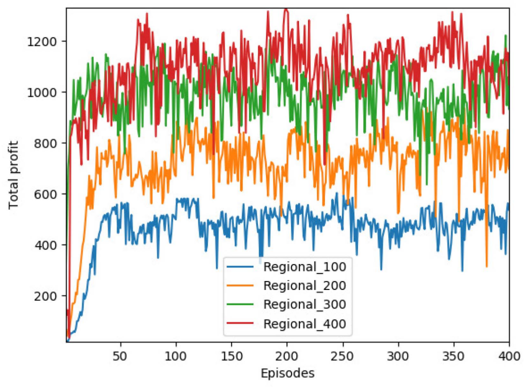
B. Performance in the Training Process

1) Convergence Analysis: In this section, the convergence of the proposed RL is tested. Fig. 5 shows the total profit of each episode in different scenarios. The total profit of each episode rises sharply at the beginning of training and then fluctuates around a certain value until the end of the training, even if the task set for training has changed. This shows that the trained Q-network is good at convergency and can be well applied to unknown scenarios.

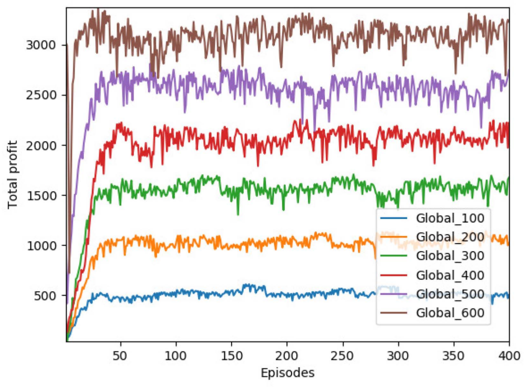
There are several reasons behind the fluctuations. The main reason is that the "exploration" in the algorithm. The purpose of exploration is to enable the agent to find better solutions, but sometimes the quality of the solution may be worse than expected. When other conditions are unchanged, the larger the exploration rate, and the greater the fluctuation.

The total profit of all scenes converge in the long run, but the performances are different for every time at the beginning of training. Fig. 6 shows the training data of the first 20 episodes in all scenarios. We saw a waving increment of the total profit in most of the scenarios, while the data in Global_600 and Regional_400 change dramatically at the first ten episodes. This is because when learning using a small number of samples, the algorithm may find a wrong value function, which leads to a wrong direction of the decision in the next one or several episodes. Nevertheless, the algorithm can always find the direction in which the Q-network converges from more experiences.

Referring to [37] and [35], we compare the performance in the convergence of our method to that of the asynchronous advantage actor–critic algorithm and actor–critic algorithm



(a)



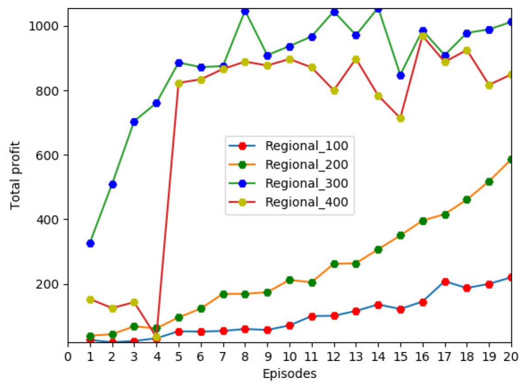
(b)

Fig. 5. Total reward of per episode in different scenarios. (a) Total profit in training process of regional distribution. (b) Total profit in training process of global distribution.

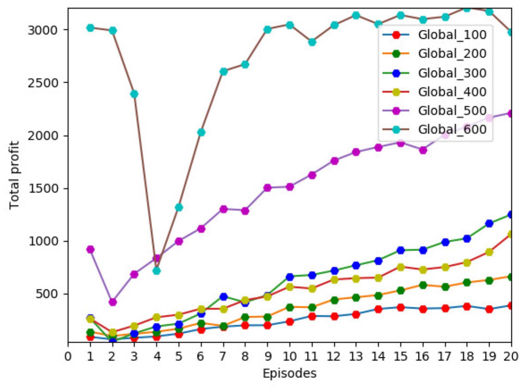
with pointer network. For controlling the variables, the same hyperparameters are applied to these three algorithms, including the learning rate, exploration rate, and the structure and the hyperparameters of the value network. We run these three algorithms 1000 times in the same scenario, and the total profit recorded in each iteration is displayed in Fig. 7. It can be found that: deep Q-learning converges fast at the beginning of the training; the total profit of asynchronous advantage actor–critic rises slowly at the first 400 episodes and grows sharply after that. The convergence accuracy of asynchronous advantage actor–critic is still lower than deep Q-learning, and the results are not as stable as deep Q-learning; Actor–Critic with pointer network maintains a low level of the total profit during the first 1000 episodes because the algorithm converges striking slowly in the first 1000 generations, or it has fallen into a local optimum.

In this part, the total profit is used to measure the convergence of our algorithm, which is the value of the objective function of the AEOS scheduling problem. The results recorded in the training process and the comparison with other RL algorithms show that the proposed deep Q-learning reaches a satisfactory level of convergence.

2) Training Time Analysis: The training time for each scenario is discussed in this section. Table III shows the time spent on pretreatment and the learning process. Due to the process of pretreatment involves some physical principles, such



(a)



(b)

Fig. 6. Total reward of the first 20 episodes in different scenarios. (a) First 20 episodes of regional distribution. (b) First 20 episodes of global distribution.

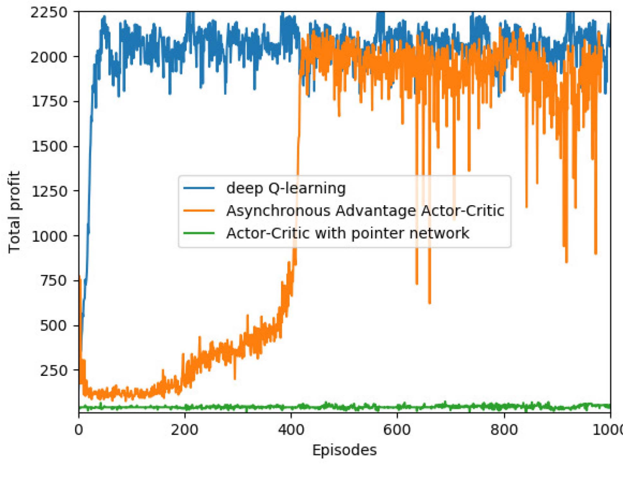


Fig. 7. Performance of deep Q-learning, asynchronous advantage actor–critic, and actor–critic with pointer network in training process.

as orbital dynamics, the calculation process is complicated, and takes a long time. The time spent on pretreatment is only related to the size of the task set. However, the time spent on learning is task-size-related and distribution-related. The longest time spent on training in these scenarios we design appears in Regional_400, which is about 3 h and 45 min. It is fully acceptable to take several hours to train a value function for an AEOS according to the capabilities and constraints of the AEOS.

TABLE III
TIME SPENT ON TRAINING PROCESS

Scenarios	Pre-treatment time	Learning time	Time spent on each data set (per 20 episodes)
Regional_100	21m52s	5m11s	15s
Regional_200	44m28s	27m43s	1m23s
Regional_300	1h6m54s	1h8m4s	3m24s
Regional_400	1h30m14s	2h15m6s	6m45s
Global_100	22m38s	4m0s	12s
Global_200	45m10s	10m12s	30s
Global_300	1h7m52s	17m23s	52s
Global_400	1h30m32s	28m6s	1m24s
Global_500	1h51m25s	38m44s	2m2s
Global_600	2h12m49s	50m27s	2m31s

C. Performance in the Testing Process

There are several advanced AEOS scheduling algorithms in literatures as far as we know: ALNS [26], [47], branch-and-bound algorithm [14], and HADRT [24]. The branch-and-bound algorithm can not solve the problem in 3600 s, which is intolerable for decision-makers in most real-world applications. We compare and discuss the results of ALNS, HADRT, and the RL approach proposed in this work.

1) Comparison of the Rate of Total Profit: The rate of total profit is one of the most important indexes for evaluating AEOS scheduling algorithms, which indicates the quality of a scheme. The objective of the AEOS scheduling problem is to maximize the total profit of the scheme. The rate of total profit is proportional to the total profit in the same task set. It can be calculated as follows:

$$pr = \frac{\text{total profit of all scheduled tasks}}{\text{total profit of all tasks in task set}} \times 100\% \qquad (17)$$

pr of ALNS, HADRT, and RL have been tested in ten different scenarios which have been described in Section V-A2, and we have tested ten times under different task sets on each scenario. *pr* of these three algorithms in different task sets are summarized in Fig. 8.

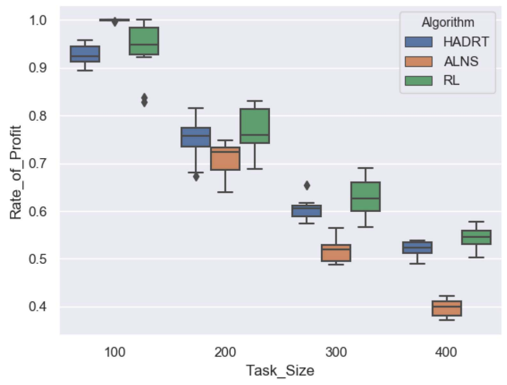
Fig. 8(a) and (b) are box diagrams to illustrate the difference in *pr* in various cases. We called scenarios in which most tasks can be scheduled (more than 90%) as under-subscribed cases, otherwise, the scenario is over-subscribed. It is easy to learn from Fig. 8 that most cases in Regional_100 and all scenarios in global distribution are under-subscribed, while all cases in Regional_200, Regional_300, and Regional_400 are over-subscribed. The average *pr* of HADRT is superior to that of ALNS in cases of over-subscribed, while the conclusion is reversed in cases of under-subscribed. The gap between results of the two algorithms are becoming larger with the task size growing.

It is worth noticing that: 1) the average *pr* of RL exceeds that of HADRT in all test scenarios and 2) RL gets the highest average *pr* in all of the over-subscribed scenarios and scenario Global_400. In other scenarios, although the average *pr* of RL does not exceed ALNS, there is only a marginal difference between the two values (less than 3%).

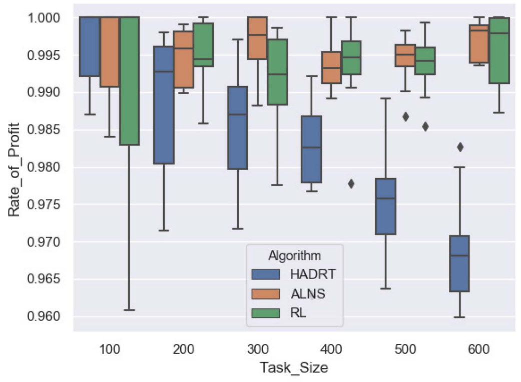
2) Comparison on CPU Time: CPU time is another crucial index to evaluate an AEOS scheduling algorithm. CPU time

TABLE IV
STATISTICS OF THE AVERAGE AND STANDARD DEVIATION OF CPU TIME

Scenarios	Average CPU time (s)			Standard deviation		
	ALNS	HADRT	RL	ALNS	HADRT	RL
Regional_100	4.37E+01	1.57E-01	3.61E-01	2.55E+01	1.61E-02	2.43E-02
Regional_200	4.46E+02	4.56E-01	1.77E+00	7.15E+01	1.70E-02	2.22E-01
Regional_300	1.39E+03	8.18E-01	3.25E+00	2.11E+02	1.63E-02	1.14E-01
Regional_400	2.75E+03	1.27E+00	4.21E+00	2.11E+02	4.01E-02	1.02E-01
Global_100	1.80E+01	6.68E-02	2.95E-01	9.27E+00	2.57E-03	3.96E-03
Global_200	8.74E+01	2.07E-01	8.97E-01	6.95E+01	7.71E-03	2.29E-02
Global_300	1.45E+02	4.12E-01	2.28E+00	1.79E+01	1.25E-02	2.59E-02
Global_400	3.87E+02	6.94E-01	3.13E+00	3.43E+02	2.22E-02	2.21E-01
Global_500	9.98E+02	1.04E+00	4.53E+00	6.30E+02	2.61E-02	7.29E-02
Global_600	1.81E+03	1.47E+00	5.87E+00	6.95E+02	2.71E-02	2.66E-02



(a)



(b)

Fig. 8. Profit rate of three methods in different test sets. (a) Profit rate of regional distribution test sets. (b) Profit rate of global distribution test sets.

of RL refers to the time consumed used in the testing process. The average and standard deviation of CPU time of these three algorithms in different task sets are recorded in Table IV.

It is obvious to note that: ALNS is always the most time-consuming algorithm, with its average CPU time for more than 40 s in Regional_100. In Regional_200, the value increases ten times of that in the condition of Regional_100, while it becomes more than 2700 s in the scenario of Regional_400. Meanwhile, a similar trend is spotted in global distribution scenarios: CPU time of ALNS grows significantly with the size of task sets. It is also noticeable that the standard deviation of ALNS is much larger than that of RL and HADRT. In a word, compared with RL and HADRT, ALNS is the worst in time efficiency.

The Q-network can be applied directly to different task sets once the training is completed. The CPU time of HADRT and RL are both at a low level and stable. Furthermore, comparing the CPU time of different distribution modes at the same task scale for one algorithm, we find the calculation time of RL and HADRT is less affected by the distribution of tasks.

The above analyses of the profit and the CPU time for different algorithms show that: HADRT performs best in terms of CPU time, but the total profit in most scenarios is less than the other two algorithms; ALNS achieves high total profit in some scenarios, but CPU time is unacceptable. Considering the total profit and CPU time comprehensively, RL can get a satisfactory solution in an acceptable time.

VI. CONCLUSION

This article creatively models a general MDP for the AEOS scheduling problem, which separates the AEOS scheduling problem to a sequencing problem and a timing problem, and the decision-making processes of these problems are handled in the agent and the environment of the MDP, respectively. State, action, reward, and the estimated long-term reward (the value in MDP) in this model are designed according to the common points of AEOS scheduling problems. And then the model is solved by combining an advanced heuristic algorithm (HADRT) with improved deep Q-learning. The deep Q-learning with an efficient pruning strategy and a novel training framework is developed to fit the AEOS scheduling problem. Experiments verify that this approach is effective in solving the AEOS scheduling problem.

Our method performs well on the satellite scheduling problem with sufficient transmission resources and electricity. However, the proposed approach may not be suitable to problem with more complex constraints and settings. In the future, the following two points are our directions: 1) to combine the achievements of RL in the field of multiagent and dynamic decision-making to explore the application of RL in a more complex environment and 2) to improve the RL algorithm to enhance its adaptability and solution performance.

ACKNOWLEDGMENT

The authors would like to thank Dr. Lei He at National University of Defense Technology for reading their article and giving suggestions.

REFERENCES

[1] W.-C. Lin, D.-Y. Liao, C.-Y. Liu, and Y.-Y. Lee, "Daily imaging scheduling of an earth observation satellite," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 35, no. 2, pp. 213–223, Mar. 2005.

[2] M. Lemaítre, G. Verfaillie, F. Jouhaud, J. M. Lachiver, and N. Bataille, "Selecting and scheduling observations of agile satellites," *Aerosp. Sci. Technol.*, vol. 6, no. 5, pp. 367–381, 2002.

[3] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 2018.

[4] D. Silver *et al.*, "Mastering the game of Go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–503, 2016.

[5] D. Silver *et al.*, "Mastering the game of Go without human knowledge," *Nature*, vol. 550, no. 7676, pp. 354–371, 2017.

[6] D. Silver *et al.*, "A general reinforcement learning algorithm that masters chess, shogi, and go through self-play," *Science*, vol. 362, no. 6419, pp. 1140–1144, 2018.

[7] G. Beaumet, G. Verfaillie, and M.-C. Charmeau, "Feasibility of autonomous decision making on board an agile earth-observing satellite," *Comput. Intell.*, vol. 27, no. 1, pp. 123–139, 2011.

[8] B. Wille, M. T. Wörle, and C. Lenzen, "VAMOS–verification of autonomous mission planning on-board a spacecraft," *IFAC Proc. Vol.*, vol. 46, no. 19, pp. 382–387, 2013.

[9] S. Peng, H. Chen, C. Du, J. Li, and N. Jing, "Onboard observation task planning for an autonomous earth observation satellite using long short-term memory," *IEEE Access*, vol. 6, pp. 65118–65129, 2018.

[10] J. Barry, "Increasing autonomy through satellite expert system scheduling," in *Proc. 2nd Space Symp.*, 1988, pp. 153–156.

[11] J. M. Barry and C. Sary, "Expert system for on-board satellite scheduling and control," NASA, Washington, DC, USA, Rep. N89-15576, 1988.

[12] W. J. Wolfe and S. E. Sorensen, "Three scheduling algorithms applied to the earth observing systems domain," *Manag. Sci.*, vol. 46, no. 1, pp. 148–166, 2000.

[13] N. Zufferey and M. Vasquez, "A generalized consistent neighborhood search for satellite range scheduling problems," *RAIRO Oper. Res.*, vol. 49, no. 1, pp. 99–121, 2015.

[14] X. Chu, Y. Chen, and L. Xing, "A branch and bound algorithm for agile earth observation satellite scheduling," *Discrete Dyn. Nat. Soc.*, vol. 2017, pp. 1–15, 2017.

[15] X. Chu, Y. Chen, and Y. Tan, "An anytime branch and bound algorithm for agile earth observation satellite onboard scheduling," *Adv. Space Res.*, vol. 60, no. 9, pp. 2077–2090, 2017.

[16] X. Hu, W. Zhu, B. An, P. Jin, and W. Xia, "A branch and price algorithm for EOS constellation imaging and downloading integrated scheduling problem," *Comput. Oper. Res.*, vol. 104, pp. 74–89, Apr. 2019.

[17] B. Bai, R. He, J. Li, and Y. Chen, "Satellite orbit task merging problem and its dynamic programming algorithm," *Syst. Eng. Electron.*, vol. 31, no. 7, pp. 1738–1742, 2009.

[18] S. A. Chien, R. Knight, A. Stechert, R. Sherwood, and G. Rabideau, "Using iterative repair to improve the responsiveness of planning and scheduling," in *Proc. AIPS*, 2000, pp. 300–307.

[19] S. Chien, D. Tran, G. Rabideau, S. Schaffer, D. Mandl, and S. Frye, "Planning operations of the earth observing satellite EO-1: Representing and reasoning with spacecraft operations constraints," in *Proc. 6th Int. Workshop Plan. Scheduling Space (IWPSS)*, 2009, pp. 1–8.

[20] S. Chien and M. Troesch, "Heuristic onboard pointing re-scheduling for an earth observing spacecraft," in *Proc. 25th Int. Conf. Autom. Plan. Scheduling (ICAPS)*, 2015, pp. 1–8.

[21] G. Beaumet, G. Verfaillie, and M.-C. Charmeau, "Decision-making onboard an autonomous agile earth-observing satellite," in *Proc. 18th Int. Conf. Autom. Plan. Scheduling (ICAPS)*, 2008, pp. 1–7.

[22] G. Beaumet, G. Verfaillie, and M.-C. Charmeau, "Autonomous planning for an agile earth-observing satellite," in *Proc. ISAIRAS*, 2008, pp. 1–6.

[23] C. Lenzen, M. T. Woerle, T. Göttfert, F. Mrowka, and M. Wickler, "Onboard planning and scheduling autonomy within the scope of the firebird mission," in *Proc. SpaceOps Conf.*, 2014, pp. 1759–1768.

[24] Y. He, Y. Chen, J. Lu, C. Chen, and G. Wu, "Scheduling multiple agile earth observation satellites with an edge computing framework and a constructive heuristic algorithm," *J. Syst. Archit.*, vol. 95, pp. 55–66, May 2019.

[25] T. Bäck, U. Hammel, and H.-P. Schwefel, "Evolutionary computation: Comments on the history and current state," *IEEE Trans. Evol. Comput.*, vol. 1, no. 1, pp. 3–17, Apr. 1997.

[26] X. Liu, G. Laporte, Y. Chen, and R. He, "An adaptive large neighborhood search metaheuristic for agile satellite scheduling with time-dependent transition time," *Comput. Oper. Res.*, vol. 86, pp. 41–53, Oct. 2017.

[27] H. Lei, L. Xiaolu, L. Gilbert, C. Yingwu, and C. Yingguo, "An improved adaptive large neighborhood search algorithm for multiple agile satellites scheduling," *Comput. Oper. Res.*, vol. 100, no. 12, pp. 12–25, 2018.

[28] F. Xhafa, J. Sun, A. Barolli, A. Biberaj, and L. Barolli, "Genetic algorithms for satellite scheduling problems," *Mobile Inf. Syst.*, vol. 8, no. 4, pp. 351–377, 2012.

[29] F. Zhang, Y. Chen, and Y. Chen, "Evolving constructive heuristics for agile earth observing satellite scheduling problem with genetic programming," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, 2018, pp. 1–7.

[30] Z. Yan and Y. Chen, "Integration schedule of agile satellite based on improved ant colony algorithm," in *Proc. IEEE Conf. Anthol.*, 2013, pp. 1–4.

[31] H. Chen, G. Wu, W. Pedrycz, P. N. Suganthan, L. Xing, and X. Zhu, "An adaptive resource allocation strategy for objective space partition-based multiobjective optimization," *IEEE Trans. Syst., Man, Cybern., Syst.*, early access, Mar. 12, 2019, doi: 10.1109/TSMC.2019.2898456.

[32] M. Moravčík *et al.*, "DeepStack: Expert-level artificial intelligence in heads-up no-limit poker," *Science*, vol. 356, no. 6337, pp. 508–513, 2017.

[33] R. Cui, C. Yang, Y. Li, and S. Sharma, "Adaptive neural network control of AUVs with control input nonlinearities using reinforcement learning," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 47, no. 6, pp. 1019–1029, Jun. 2017.

[34] C. Wang, J. Wang, Y. Shen, and X. Zhang, "Autonomous navigation of UAVs in large-scale complex environments: A deep reinforcement learning approach," *IEEE Trans. Veh. Technol.*, vol. 68, no. 3, pp. 2124–2136, Mar. 2019.

[35] M. Nazari, A. Oroojlooy, L. Snyder, and M. Takác, "Reinforcement learning for solving the vehicle routing problem," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 9839–9849.

[36] W. Usaha and J. A. Barria, "Reinforcement learning for resource allocation in LEO satellite networks," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 37, no. 3, pp. 515–527, Jun. 2007.

[37] W. Haijiao, Y. Zhen, Z. Wugen, and L. Dalin, "Online scheduling of image satellites based on neural networks and deep reinforcement learning," *Chin. J. Aeronautics*, vol. 32, no. 4, pp. 1011–1019, 2019.

[38] L. Wang, C. Jiang, L. Kuang, S. Wu, L. Fei, and H. Huang, "Mission scheduling in space network with antenna dynamic setup times," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 55, no. 1, pp. 31–45, Feb. 2019.

[39] L. Wang, C. Jiang, L. Kuang, S. Wu, H. Huang, and Y. Qian, "High-efficient resource allocation in data relay satellite systems with users behavior coordination," *IEEE Trans. Veh. Technol.*, vol. 67, no. 12, pp. 12072–12085, Dec. 2018.

[40] L. Barbulescu, A. Howe, and D. Whitley, "AFSCN scheduling: How the problem and solution have evolved," *Math. Comput. Modell.*, vol. 43, nos. 9–10, pp. 1023–1037, 2006.

[41] K. Sun, S. Mou, J. Qiu, T. Wang, and H. Gao, "Adaptive fuzzy control for nontriangular structural stochastic switched nonlinear systems with full state constraints," *IEEE Trans. Fuzzy Syst.*, vol. 27, no. 8, pp. 1587–1601, Aug. 2019.

[42] J. Qiu, K. Sun, I. J. Rudas, and H. Gao, "Command filter-based adaptive NN control for MIMO nonlinear systems with full-state constraints and actuator hysteresis," *IEEE Trans. Cybern.*, vol. 50, no. 7, pp. 2905–2915, Jul. 2020.

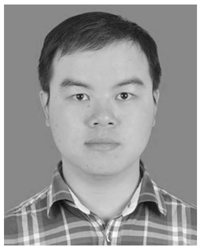
[43] A. K. Sadhu and A. Konar, "Improving the speed of convergence of multi-agent Q-learning for cooperative task-planning by a robot-team," *Robot. Auton. Syst.*, vol. 92, pp. 66–80, Jun. 2017.

[44] S. Adam, L. Busoniu, and R. Babuska, "Experience replay for real-time reinforcement learning control," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 42, no. 2, pp. 201–212, Mar. 2012.

[45] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," in *Proc. Int. Conf. Learn. Represent.*, 2016, pp. 1–23.

[46] S. A. Kumar and K. Amit, "An efficient computing of correlated equilibrium for cooperative Q-learning-based multi-robot planning," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 50, no. 8, pp. 2779–2794, Aug. 2020.

[47] L. He, M. de Weerdt, and N. Yorke-Smith, "Time/sequence-dependent scheduling: The design and evaluation of a general purpose tabu-based adaptive large neighbourhood search algorithm," *J. Intell. Manuf.*, vol. 31, no. 4, pp. 1051–1078, 2020.



Yongming He received the B.S. degree in logistics engineering from Chang'an University, Xi'an, China, in 2014. He is currently pursuing the Ph.D. degree in management science and engineering with the College of Systems Engineering, National University of Defense Technology, Changsha, China.

He was a visiting Ph.D. student with the University of Alberta, Edmonton, AB, Canada, from November 2018 to November 2019. His research interests include operations research, artificial intelligence, intelligent decision, task scheduling, and planning.

Lining Xing received the bachelor's degrees in economics and in science from Xi'an Jiaotong University, Xi'an, China, in 2002, and the Ph.D. degree in management science from the National University of Defense Technology, Changsha, China, in 2009.

He visited the School of Computer, University of Birmingham, Birmingham, U.K., from November 2007 to November 2008. He is a Professor with the School of Systems Engineering, National University of Defense Technology. His research interests include intelligent optimization methods and scheduling of resources.

Yingwu Chen received the B.S. degree in automation, the M.S. degree in system engineering, and the Ph.D. degree in engineering from the National University of Defense Technology (NUDT), Changsha, China, in 1984, 1987, and 1994, respectively.

He was a Lecturer from 1989 to 1994 and an Associate Professor from 1994 to 1999 with NUDT, where he was a Professor from 1999 to 2017 and the Director of the Department of Management Science and Engineering, College of Information Systems and Management. He has been a Distinguished Professor with the College of Systems Engineering, NUDT, where he focuses on management theory and its applications. He has authored more than 70 research publications. His current research interests include assistant decision-making systems for planning, decision-making systems for project evaluation, management decisions, and artificial intelligence.

Dr. Chen is an Editor of the *Principle of System Engineering* (Press of National University of Defense Technology) and the *Technology of Quantificational Analysis* (China Renmin University Press).

Witold Pedrycz (Life Fellow, IEEE) received the M.Sc., Ph.D., and D.Sc. degrees in computer science from the Silesian University of Technology, Gliwice, Poland, in 1977, 1980, and 1984, respectively.

He is a Professor and the Canada Research Chair of computational intelligence with the Department of Electrical and Computer Engineering, University of Alberta, Edmonton, AB, Canada. He is also with the Systems Research Institute, Polish Academy of Sciences, Warsaw, Poland. He has also authored 18 research monographs and edited volumes covering various aspects of *Computational Intelligence*, *Data Mining*, and *Software Engineering*. His main research directions involve computational intelligence, fuzzy modeling and granular computing, knowledge discovery and data science, pattern recognition, data science, knowledge-based neural networks, and control engineering. He has published numerous papers in the above areas; the current H-index is 114 (Google Scholar).

Prof. Pedrycz is a recipient of the IEEE Canada Computer Engineering Medal, the Cajastur Prize for Soft Computing from the European Centre for Soft Computing, the Killam Prize, the Fuzzy Pioneer Award from the IEEE Computational Intelligence Society, and the Prestigious Norbert Wiener Award from the IEEE Systems, Man, and Cybernetics Society in 2007. He is vigorously involved in editorial activities. He is the Editor-in-Chief of *Information Sciences* and *WIREs Data Mining and Knowledge Discovery* (Wiley), and the Co-Editor-in-Chief of the *International Journal of Granular Computing* (Springer) and *Journal of Data, Information and Management* (Springer). He serves on an Advisory Board of the IEEE TRANSACTIONS ON FUZZY SYSTEMS and is a member of a number of editorial boards of international journals. In 2009, he was elected a Foreign Member of the Polish Academy of Sciences. In 2012, he was elected a Fellow of the Royal Society of Canada.

Ling Wang received the B.Sc. degree in automation and the Ph.D. degree in control theory and control engineering from Tsinghua University, Beijing, China, in 1995 and 1999, respectively.

Since 1999, he has been with the Department of Automation, Tsinghua University, where he became a Full Professor in 2008. He has authored five academic books and over 260 refereed papers. His current research interests include intelligent optimization and production scheduling.

Prof. Wang was a recipient of the National Natural Science Fund for Distinguished Young Scholars of China, the National Natural Science Award (Second Place) in 2014, the Science and Technology Award of Beijing City in 2008, the Natural Science Award (First Place in 2003 and Second Place in 2007) nominated by the Ministry of Education of China. He is currently the Editor-in-Chief of the *International Journal of Automation and Control* and an Associate Editor of the IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION.

Guohua Wu received the B.S. degree in information systems and the Ph.D. degree in operations research from the National University of Defense Technology, Changsha, China, in 2008 and 2014, respectively.

From 2012 to 2014, he was a visiting Ph.D student with the University of Alberta, Edmonton, AB, Canada. He is currently a Professor with the School of Traffic and Transportation Engineering, Central South University, Changsha. He has authored more than 60 referred papers, including those published in the IEEE TRANSACTIONS ON CYBERNETICS, IEEE TRANSACTIONS ON SYSTEMS, MAN AND CYBERNETICS, *Information Sciences*, and *Computers & Operations Research*. His current research interests include planning and scheduling, computational intelligence, and machine learning.

Prof. Wu serves as an Associate Editor of the *Swarm and Evolutionary Computation Journal*, an Editorial Board Member of the *International Journal of Bio-Inspired Computation*, and the Guest Editor of *Information Sciences* and *Memetic Computing*. He is a Regular Reviewer of more than 20 journals, including the IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION, IEEE TRANSACTIONS ON CYBERNETICS, *Information Sciences*, and *Applied Soft Computing*.