Learning Adaptive Genetic Algorithm for Earth Electromagnetic Satellite Scheduling

Yanjie Song
Xidian University, China
National University of Defense Technology, China

Junwei Ou
National University of Defense Technology, China

Ponnuthurai Nagaratnam Suganthan, Fellow, IEEE
Qatar University, Qatar

Witold Pedrycz, Life Fellow, IEEE
University of Alberta, Canada
Polish Academy of Sciences, Poland
Department of Computer Engineering, Sariyer/Istanbul, Turkiye

Qinwen Yang
The University of Auckland, New Zealand

Lining Xing
Xidian University, China

***Abstract*— Earth electromagnetic exploration satellites are widely used in many fields due to their wide detection range and high detection sensitivity. The complex environment and the proliferating number of satellites make management a primary issue. In this paper, a learning adaptive genetic algorithm (LAGA) is proposed for the Earth electromagnetic satellite scheduling problem (EESSP). Control parameters are essential to the successful performance of evolutionary algorithms, and their sensitivity to the problem makes tuning parameters very time-consuming. In the LAGA, a gated recurrent unit (GRU) neural network model is used to control the parameters of variation operators. The neural network model is capable of leveraging real-time information to achieve dynamic parameter adjustment during population search. Moreover, a policy gradient-based reinforcement learning method is utilized to update the parameters of GRU. An adaptive evolution mechanism is employed in LAGA for the autonomous selection of crossover operators. Additionally, the heuristic initialization method, elite strategy, and local search method are incorporated into LAGA to enhance overall performance. Simulation experiments demonstrate the effectiveness of LAGA in solving the EESSP. This study highlights the advantages of utilizing reinforcement learning to optimize neural network models for controlling genetic algorithm searches. Learning adaptive planning methods can effectively address complex problem scenarios and enhance satellite scheduling system performance.**

***Index Terms*— reinforcement learning, learning adaptive, control parameters, earth electromagnetic satellite scheduling, genetic algorithm, GRU**

Manuscript received XXXXX 00, 0000; revised XXXXX 00, 0000; accepted XXXXX 00, 0000.
This work is supported by the Science and Technology Innovation Team of Shaanxi Province (2023-CX-TD-07), the Special Project in Major Fields of Guangdong Universities (2021ZDZX1019), and the Hunan Key Laboratory of Intelligent Decision-making Technology for Emergency Management (2020TP1013).

Authors' addresses: Yanjie Song is with the School of Electronic Engineering, Xidian University, China; College of Systems Engineering, National University of Defense Technology, China (e-mail:songyj_2017@163.com). Junwei Ou is with the College of Systems Engineering, National University of Defense Technology, China (e-mail:junweiou@163.com). P. N. Suganthan is with the KINDI Center for Computing Research, College of Engineering, Qatar University, Doha, Qatar (e-mail:p.n.suganthan@qu.edu.qa). Witold Pedrycz is with the Department of Electrical and Computer Engineering, University of Alberta, Canada; Systems Research Institute, Polish Academy of Sciences, Poland; Faculty of Engineering and Natural Sciences, Department of Computer Engineering, Sariyer/Istanbul, Turkiye (e-mail:wpedrycz@ualberta.ca). Qinwen Yang is with School of Computer Science, The University of Auckland, New Zealand (e-mail:yangqq7160@gmail.com). Lining Xing is with the School of Electronic Engineering, Xidian University, China (e-mail:lnxing@xidian.edu.cn).(Corresponding author: Lining Xing)
*Yanjie Song and Junwei Ou contributed equally to this article.

I. Introduction

In recent years, the rapid development of satellite technology has significantly transformed our lives. However, the surge in user demands and the proliferation of satellites have posed significant challenges to managing Earth's electromagnetic satellite (EES) resources [1], [2]. EES is an artificial earth satellite equipped with antennas and signal-receiving equipment that can acquire surface electromagnetic signal data. The ability to meet users' diverse detection requirements and promptly obtain the corresponding data is crucial for satellite control. The Earth Electromagnetic Satellite Scheduling Problem (EESSP) is proposed in response to the challenges faced in managing satellites. Specifically, the EESSP aims to obtain optimal plans for a series of EESs while satisfying various constraints. As one type of satellite task scheduling problems, the EESSP necessitates meticulous model construction and intelligent algorithm design to achieve satisfactory plans.

In the EESSP, a series of EESs and tasks need to be scheduled. Each satellite flies in a fixed orbit, limiting its coverage area. The detection range of the satellite antenna is determined by its aperture size at design time. Only tasks within an EES's detection range can be detected [3]. In addition, the number of tasks that can be accomplished is severely limited by several other settings and satellite operating conditions. Therefore, this study proposes a mixed integer programming (MIP) model and a learning adaptive genetic algorithm (LAGA) to solve EESSP.

Among the existing studies on the satellite scheduling problem, the EESSP is still in its early stages of development, while the optical satellite scheduling problem (OSSP) has been extensively investigated [4], [5]. Both types of problems fall under the Earth Observation Satellite Scheduling Problem (EOSSP). While the EOSSP does

not require consideration of cloud cover, resolution, and other factors like the OSSP, it necessitates a set of constraints closely linked to electromagnetic detection such as detection mode, bandwidth, and polarization mode [4]. While there exist certain distinctions between these two question types, the proposed model and method employed to address the OSSP serves as a valuable reference for the EESSP.

Recent studies have emerged a series of good models and effective solutions for addressing the EOSSP. The MIP model [6] and the constraint satisfaction problem (CSP) model [7] are the two most common forms of models. The CSP model aims to find a feasible solution rather than the optimal solution. However, the goal of our work is to obtain the optimal or approximate optimal solution, thus, to this end, MIP is more suitable for the studied problem. Detailed research progress about the EOSSP model is described in [8]. A series of exact solution algorithms and approximate solution algorithms are proposed for specific planning problems. The solving algorithms for EOSSP can be divided into two categories: exact solution algorithms and approximate solution algorithms. Among them, the exact solution algorithm can find the optimal solution or boundary value of the problem. However, for the large-scale problem situations we are faced in practical applications, the exact algorithm has the drawback of the exponential explosion of computation time, which limits its widespread use. Unlike exact solution algorithms, approximate solution algorithms (including heuristic algorithms, metaheuristics, etc.) are not guaranteed to find the optimal solution. The main advantage of such algorithms is to obtain an acceptable solution in a short time. Further, the search performance can be improved by improvements in algorithmic strategies, search patterns, etc. Among the metaheuristic algorithms, genetic algorithm (GA) [9], ant colony algorithm [10], memetic algorithm [11], and others have been successfully tried to solve all kinds of EOSSP problems. In this study, LAGA is elaborately designed based on the framework of Genetic Algorithm (GA) to cater to the characteristics of EESSP. The encoding and decoding mechanism of GA can be seamlessly integrated with the task sequencing process. Moreover, GA exhibits remarkable global search capability and convergence performance. In these relevant studies, it is crucial to consider the balance between exploration and exploitation in algorithm design. Among them, machine learning methods can enhance the adaptability of metaheuristic algorithms to different problem scenarios through data-driven training and learning [12], [13]. This paper utilizes reinforcement learning (RL) to enhance the algorithm based on the traditional genetic algorithm (GA). In RL, the agent optimizes the parameters of the gated recurrent unit (GRU) neural network model based on the attributes associated with EESSP. Subsequently, the trained GRU model is employed for dynamic adjustment of the algorithm search.

Strategies for adjusting GA search modes are diverse. In this study, we employ a strategy of adjusting control parameters to enhance the algorithm's search performance. The precise adjustment of control parameters governing crossover and mutation plays a pivotal role in achieving optimal solutions within the framework of GA. In general, the crossover probability (denoted as CR) of GA is in the interval of [0.7,0.99], and the mutation probability (denoted as MR) is in the interval of [0.01,0.3] [14]. In the process of solving a combinatorial optimization problem, an effective algorithm should initially explore as many new solution spaces as possible. However, after several generations of evolution, the algorithm should shift towards exploiting a smaller solution space. The optimal values for these two control parameters vary depending on the specific problem scenario and can be challenging to determine for each instance of EESSP. To address this situation, a novel approach is proposed by integrating reinforcement learning with a neural network model to control the parameters. This parameter control method enhances the genetic algorithm (GA) and introduces an adaptive learning capability named LAGA. LAGA aims to minimize the cost associated with parameter tuning while finding the optimal solution for EESSP. The proposed algorithm employs a Gated Recurrent Unit (GRU) neural network model to determine the control parameters, which are trained using the reinforcement learning (RL) method. Features related to the EESSP problem are utilized for designing the agent's properties in RL. Following training, the GRU can acquire rational control parameters that facilitate the search process of LAGA. Furthermore, initialization methods associated with EESSP and evolutionary operations are also incorporated into the algorithm. The main contributions of this paper are as follows.

1. A mixed-integer programming model is formulated to address the EESSP problem, aiming at maximizing the detection profit of task scheduling. The model incorporates constraints on satellite capabilities and task execution requirements, while transition times between two tasks are treated as maximization functions. By refining the modeling with practical parameters such as detection mode and bandwidth guarantees, this approach demonstrates its potential for practical applications.

2. An evolutionary algorithm utilizing adaptive learning is proposed, which treats the population evolution process as a time series and employs a GRU model to derive control parameters for the evolution operation based on online information prediction. A policy gradient-based reinforcement learning training method is presented to optimize the GRU parameters. Additionally, in LAGA, a heuristic initialization method is designed to generate high-quality initial populations. The adaptive crossover mechanism is employed to achieve efficient population exploration, while an elite strategy and a local search method are utilized to expedite the convergence of the iterative algorithm. Experimental results demonstrate that the proposed approach can yield plans with high detection profit.

The remainder of the paper is organized as follows. Section II introduces the related work of this study.

Section III introduces the description of EESSP and the mathematical model. Section IV introduces the genetic algorithm and reinforcement learning methods based on adaptive learning. Section V verifies the performance of the proposed algorithm through several experiments. Section VI summarizes the conclusions obtained from the study and introduces future research directions.

II. Related Work

The mixed-integer programming model is a classical model form for constructing mathematical models of EOSSP in many studies [15], [16], [17]. Chen et al. proposed a conflict metric and analyzed the interdependence of time windows to construct a mixed-integer programming model [15]. Zhu et al. used a directed acyclic graph to represent feasible observation task plans [16]. Valicka et al. proposed an extended two-stage and three-stage stochastic mixed-integer scheduling model by considering cloud uncertainty [17]. Some other model forms, such as quadratic scheduling models and graphical models, have also been used by researchers [18], [19].

The solution algorithm is also vital for solving the EOSSP problem. Among many EAs, GA has strong applicability and good global search performance. [20], [21], [22]. Chen et al. took into consideration the significant computational cost associated with satellite task scheduling problems and devised a population perturbation mechanism within a genetic algorithm. This mechanism serves to enhance the algorithm's capability in locating the optimal solution [23]. Li et al. used a new encoding method in a genetic algorithm [24]. Individual coding is utilized to determine, based on the ground station ID, a reduction in computational complexity during task scheduling. The applicability of algorithms to various scenarios has been the focus of research. To our knowledge, there does not exist any evolutionary algorithm with a parameter control strategy to solve the EOSSP. Reasonable parameter control can enhance the generalization ability of the algorithm.

Besides EA, the reinforcement learning method has also emerged as a viable approach for tackling intricate optimization problems. Huang et al., He et al., Lam et al., and Ren et al. respectively employed RL to derive highly efficient satellite observation plans [25], [26], [27], [28]. Huang et al. treated the EOSSP problem as a Markov decision process in continuous time and constructed a reinforcement learning algorithm based on policy gradient [25]. He et al. employed a Markov decision process for the completion of observation tasks assignment, followed by utilizing a dynamic scheduling approach to derive a specific execution plan [26]. Lam et al. learned a heuristic algorithm structure by reinforcement learning to achieve that some subsequent tasks can be selected after a given part of the task solution [27]. Ren et al. designed a block encoding reinforcement learning training algorithm to solve the Agile EOSSP [28]. A significant amount of training and generalizability emerge as crucial factors that impose limitations on the application of RL. Overcoming these limitations and maximizing the algorithm's utility becomes pivotal in designing an effective algorithm. Consequently, hybridizing reinforcement learning with evolutionary algorithms presents a novel approach that effectively integrates the respective advantages of both methods [29], [30], [31]. Although there have been many successful practices of combining reinforcement learning with evolutionary algorithms in numerical optimization problems [32], [33] and combinatorial optimization problems [34], [35], [36], few studies have been done to solve satellite task scheduling problems using this idea [37]. Song et al. proposed a reinforcement learning-based genetic algorithm to solve the electromagnetic detection satellite scheduling problem [38]. The proposed algorithm uses a reinforcement learning method to select crossover operators.

It is apparent from the relevant literature that there exists a paucity of research on addressing EESSP problems. Furthermore, to our knowledge, no studies have been conducted on utilizing parameter control strategies in algorithms to optimize performance. In this paper, we will establish a mathematical model and propose a learning adaptive genetic algorithm for solving the EESSP problem.

III. Model

A. Problem Description

Within a given scheduling horizon, a series of Earth Exploration Satellites (EES) are required to develop task plans for each satellite to accomplish the detection tasks proposed by users. The magnitude of these tasks far exceeds the capacity of all satellites, resulting in an oversubscription issue. Additionally, the plan needs to take into account the working capabilities, task requirements, and other circumstances of EESs.

Each task execution must meet the users' requirements, including the geographic location of the target to be detected, detection time duration, and specific time range. It is important to note that signal detection is subject to satellite visibility windows and can only occur during these periods. Furthermore, detection operations must be carried out within the designated time frame and duration. As the satellite moves through its detectable range, the angle between the antenna and the task changes. If this angle becomes too large, successful completion of the task's detection will not be possible.

After completing a detection task, an EES cannot immediately proceed to another one. The satellite must wait for the payload to undergo a series of configuration adjustments to meet the requirements of the next detection task parameters. These adjustments include changes to the detection mode, bandwidth, frequency band, and other parameters. The adjustment of parameter configuration will render a portion of the time window resources unavailable, thereby exacerbating resource scarcity and

increasing the complexity associated with plan development.

B. Symbols and Variables

This section introduces the variables and symbols involved in the mathematical model.

Sat :the set of detection satellites, $N_s = |Sat|$, s_i denotes satellite i;

O_i: the set of orbits of the satellite i;

ϑ_i: the maximum angle that can be detected by the satellite i;

T: the set of detection tasks, $N_t = |T|$, $task_j$ denotes the detection task j;

d_j: required detection time length of the task j;

$[rest_j, rlet_j]$: allowable detection time range required for task j;

$\theta_j^{\max}$: the maximum allowable detection angle for task j;

p_j: the profit that can be obtained from the successful completion of task j;

TW: the set of time windows, $N_{tw} = |TW|$;

$[evt_{ijko}, lvt_{ijko}]$: the time window k of the task j in on orbit o for satellite i;

t: the moment of satellite flight;

θ_{ij}^t: angle between the satellite's antenna and task j at moment t;

F_i^m: function for the detection mode transition time of satellite i;

F_i^b: function for the bandwidth setting transition time of satellite i;

F_i^f: function for the frequency setting transition time of satellite i;

$tr_{ijj'}^m$: detection mode transition time of satellite i between task j and task j;

$tr_{ijj'}^b$: bandwidth mode transition time of satellite i between task j and task j';

$tr_{ijj'}^f$: frequency transition time of satellite i between task j and task j';

$tr_{ijj'}$: transition time of satellite i between task j and task j';

I: a very large integer;

Decision variables:

x_{ijko}: whether satellite i performs task j within kth time window on orbit o, if it is done, $x_{ijko} = 1$; otherwise, $x_{ijko} = 0$;

st_{ijo}: the start time of the satellite i to perform the task j on orbit o.

C. Mathematical Model

In this section, a mathematical model is constructed. First, the assumptions of the model are introduced. Based on assumptions made in [38], several specific assumptions for EESSP are given as follows.

Assumptions:

1. The task is covered by a single satellite detection, without the need for multiple repetitions.

2. The detection task can be carried out at most once, without considering multiple repetitions.

3. The impact of satellite sequestration and energy on satellite detection activities is not considered.

4. Equal value of the profit obtained by the satellite from performing the task at any moment in the time window.

5. The detection tasks to be performed by the satellite are predetermined before scheduling, and there will be no adjustment of task performance requirements during both the scheduling and execution phases. This includes early or late completion of task requirements as well as temporary cancellation of tasks.

6. The satellite is capable of maintaining normal operations throughout the entire planning time horizon.

The calculation method for the transition time of satellite i between tasks j and j' is given at first. It can be calculated as follows:

$$tr_{ijj'}^m = F_i^m\left(r_j^m, r_{j'}^m\right) \tag{1}$$

$$tr_{ijj'}^b = F_i^b\left(r_j^b, r_{j'}^b\right) \tag{2}$$

$$tr_{ijj'}^f = F_i^f\left(r_j^f, r_{j'}^f\right) \tag{3}$$

$$tr_{ijj'} = \max\left\{0, tr_{ijj'}^m, tr_{ijj'}^b, tr_{ijj'}^f\right\} \tag{4}$$

where r_j^m, r_j^b, r_j^f denote detection mode setting requirement, bandwidth setting required for a task, and frequency setting requirement for task j respectively. The maximum time required to convert satellite parameters is defined as the transition time.

Objective function:

We aim to identify a sequence of tasks that can be executed within the solution space, yielding high detection profits. Therefore, our objective function seeks to maximize the profit generated by the task plan.

$$\sum_{i\in S}\sum_{j\in J}\sum_{k\in TW}\sum_{o\in O_i} p_j \cdot x_{ijko} \tag{5}$$

where p_j denotes the profit that can be obtained by the successful completion of task j, x_{ijko} denotes whether satellite i performs task j within kth time window of orbit o.

Constraints:

$$st_{ijo} \le rest_j \cdot x_{ijko}, i \in Sat, j \in T, k \in TW, o \in O_i \tag{6}$$

$$(st_{ijo} + d_j)\cdot x_{ijko} \le rlet_j, i \in Sat, j \in T, k \in TW, o \in O_i \tag{7}$$

$$st_{ijo} \le evt_{ijko} \cdot x_{ijko}, i \in Sat, j \in T, k \in TW, o \in O_i \tag{8}$$

$$(st_{ijo} + d_j)\cdot x_{ijko} \le evt_{ijko}, i \in Sat, j \in T, k \in TW, o \in O_i \tag{9}$$

$$\begin{aligned}\theta_{ij}^t \cdot x_{ijko} \le \min\left\{\vartheta_i, \theta_j^{\max}\right\}, i \in Sat, j \in T \\ , k \in TW, o \in O_i, t \in [st_{ijo}, st_{ijo} + dur_j]\end{aligned} \tag{10}$$

$$\sum_{i\in S}\sum_{k\in TW}\sum_{o\in O_i} x_{ijko} \le 1, i \in Sat, j \in T, k \in TW, o \in O_i \tag{11}$$

$$(st_{ijo} + dur_j) \cdot x_{ijko} + tr_{ijj'} \leq st_{ij'o} + I \cdot (1 - x_{ij'k'o}), \quad j \neq j', i \in Sat, j, j' \in T, o \in O_i, k, k' \in TW \tag{12}$$

$$x_{ijko} \in \{0, 1\}, i \in Sat, j \in T, k \in TW, o \in O_i \tag{13}$$

$$st_{ijo} \in N, i \in Sat, j \in T, k \in TW, o \in O_i \tag{14}$$

Constraints (6) and (7) indicate that the task is to be executed within the required time range. Constraints (8) and (9) indicate that the task needs to be executed within the time window in which it can be detected. Constraints (10) indicate that the angle between the satellite detection and the task needs to be less than the maximum allowable angular requirement. Constraints (11) indicate that each task can only be detected at most once. Constraints (12) indicate that the transition between two tasks needs to meet the time requirement. Constraints (13) and (14) ensure that decision variables need to be valued in the corresponding ranges.

IV. The Proposed Method

A. Embedding GRU in the GA Framework

The crossover and mutation in population evolution can be considered as a stochastic time series with evident time-dependent characteristics. The control parameters influence the manner and likelihood of population evolution. Hence, these two operators play a pivotal role in determining whether the algorithm can attain the optimal solution. Therefore, a nonlinear equation prediction approach is employed to obtain parameter configurations that align with the search pattern. Alternatively, the variations in control parameters throughout population evolution can be viewed as a time series. For addressing time series-related problems, the recurrent neural network stands as the primary choice. The population evolution information encompasses both current and historical data. It is expected that the network model should allocate greater attention to the information of recent evolutionary generations while diminishing its focus on earlier temporal evolutions. Based on the time-dependent nature of online information, a GRU model is utilized to extract valuable insights that can aid in GA search. The GRU model, proposed by Cho, is a classical recurrent neural network (RNN) that effectively addresses issues related to long-term memory dependence and gradient explosion when compared with traditional RNN models [39], [40]. Moreover, the GRU model requires fewer parameters and fewer training times compared to the classical long-short memory network (LSTM) model [41]. Each GRU model is composed of a series of GRU units, which effectively capture the interrelationships among data in the temporal dimension through unit combination. The gate structure is a unique information flow regulation mechanism for LSTM and GRU models, as the memory gate information is omitted from the GRU cell structure. To describe GRU more intuitively, its specific structure is shown in Figure 1.

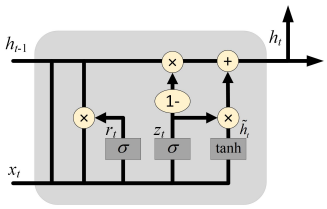


Fig. 1: GRU unit structure

In each GRU unit, the update gate and reset gate are used to achieve a good prediction of the time series. The following equations can describe the GRU.

$$h_t^j = \left(1 - z_t^j\right) h_{t-1}^j + z_t^j \tilde{h}_t^j \tag{15}$$

$$z_t^j = \sigma(W_z \mathbf{x}_t + U_z \mathbf{h}_{t-1})^j \tag{16}$$

$$\tilde{h}_t^j = \tanh\left(W \mathbf{x}_t + U\left(\mathbf{r}_t \odot \mathbf{h}_{t-1}\right)\right)^j \tag{17}$$

$$r_t^j = \sigma(W_r \mathbf{x}_t + U_r \mathbf{h}_{t-1})^j \tag{18}$$

$$\sigma(z) = \frac{1}{1 + e^{-z}} \tag{19}$$

$$\tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}} \tag{20}$$

where vector $\mathbf{h}_{t-1}$ and vector $\mathbf{x}_t$ denote the input of GRU, where $\mathbf{h}_{t-1} \in \mathbb{R}^h$, $\mathbf{x}_t \in \mathbb{R}^d$, h is the hidden layer size and d is the dimension of the feature. U denotes the input into a hidden space, $\mathbf{W}$ denotes the weight matrix, z_t^j denotes the update gate, r_t^j denotes a set of reset gates, $\odot$ denotes the multiplication of the corresponding element positions of the matrix, $\sigma(\cdot)$ denotes the sigmoid activation function, and $\tanh(\cdot)$ is the tanh function.

In practical applications of the GRU model, the complete network architecture comprises two types of neural network structures: the GRU unit and a fully connected network. The fully connected network is utilized after the GRU unit to further process data streams. The complete data stream for the GRU network model is as follows: First, the input data is processed by several GRU units. Then, several fully connected network layers are used to further process the data. Finally, the Softmax function is used to obtain the outputs.

To streamline the intermediary steps, GRU can be abbreviated as:

$$H_t = GRU(S_t, H_{t-1}, W_G) \tag{21}$$

where W_G denotes the parameter of GRU units. The fully connected network layer can be described as:

$$CR_t = Linear(H_t, W_c, b_c) \tag{22}$$

$$MR_t = Linear(H_t, W_m, b_m) \tag{23}$$

where W_c, W_m denotes network parameters and b_c, b_m denotes bias.

According to the above network model, the current state value S_t is processed as input data to fit a combination of control parameters that will help the population search. We denote $\Omega_t = [CR_t, MR_t]$ and use such parameters for the population evolution of the generation

t. Once a set of parameters has been obtained, the solution space can be explored and analyzed using these parameters. The complete data flow can be expressed as follows:

$$\Omega_t, H_t = GRU(S_t, H_{t-1}, \mathbf{W}) \tag{24}$$

After obtaining the complete data flow of the GRU model, the process of embedding the GRU model in the evolution process of the GA generation population can be given. As shown in Figure 2, the symbol S in the figure represents the selection operator, FE represents the fitness evaluation, CO represents the crossover operator, and MO represents the mutation operator. The state information S_t is the input of the GRU model. The crossover probability CR_t and the mutation probability MR_t required for the current population evolution are obtained through the output action of the neural network. These two probabilities can exert a significant impact on population exploration.

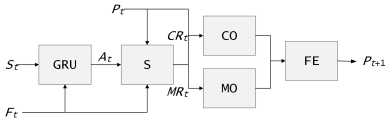


Fig. 2: GRU embedded GA in generation t

B. Reinforcement Learning for Evolutionary Search

The proposed approach employs reinforcement learning to enable the GRU model to update its network parameters based on the population evolution process, resulting in a parameter configuration that is more conducive to effective population search. Specifically, the algorithm's population evolution process is formulated as a Markov decision process (MDP). The RL method enables the agent to select the appropriate action for maximizing rewards in a dynamic environment. State, action, reward, and transition are the fundamental components of the RL approach. Each element of MDP will be comprehensively introduced in this section, followed by an overview of the policy gradient training method.

1. State

The state serves as the input to the GRU network model, enabling the agent to make informed decisions regarding action selection for the evolutionary process of LAGA. To provide a comprehensive set of features and information for optimal decision-making, it is crucial that the state accurately reflects the current population evolution dynamics. In our proposed algorithm, the set of states $\boldsymbol{S}$ denotes a set of states constitutes. $\boldsymbol{S}$ can be described as follows.

$$\boldsymbol{S} = \{S_0, S_1, ..., S_t, ...\} \tag{25}$$

where S_t denotes the agent state value at time step t. S_t consists of a set of representations of task attributes and population evolution information. The details of the specific attribute values of the state are as follows.

$$S_t = \left\{ y_j^t = (d_j, p_j, ts_j, l_j) \,|j = 1, 2, ..., N_t \right\} \tag{26}$$

where d_j is the task duration, p_j is the task profit, l_j denotes the cumulative arrangement of the task j in the current population, and its value indicates the number of successful placements. ts_j denotes the range of the indicated time interval, which can be calculated as follows:

$$ts_j = rlet_j - rest_j \tag{27}$$

2. Action

Action is a fundamental component of the MDP. The actions in reinforcement learning exhibit distinct characteristics in continuous and discrete action spaces. As the reinforcement learning method involves selecting actions within a continuous action space. Therefore, the probability density function of action A_t under state S_t based on the parameters θ of the policy is obtained. Assuming that the action probability values obey a Gaussian distribution, the policy is expressed as follows.

$$\begin{aligned} \pi\left(A_t \mid S_t\right) &= \mathcal{N}\left(A_t \mid \mathrm{GRU}\left(S_t; \mathbf{W}\right), \sigma^2\right) \\ &= \frac{1}{\sqrt{2\pi}\sigma\left(S_t; \tilde{\theta}\right)} \exp\left\{ -\frac{\left(A_t - \mu\left(S_t; \tilde{\theta}\right)\right)^2}{2\sigma^2\left(S_t; \tilde{\theta}\right)} \right\} \end{aligned} \tag{28}$$

where $\mathbf{W}$ denotes the parameter of the GRU model and $\tilde{\theta} = \left[\tilde{\theta}_\mu, \tilde{\theta}_\sigma\right]^{\mathrm{T}}$ is the parameter of the strategy, which is obtained by fitting the neural network model.

The probability density function corresponding to the action is derived from the input state. This obtained function conforms to the two control parameters, and by sampling according to this probability density function, the required values of control parameters can be obtained.

3. Reward

In our study, the reward should reflect the impact of population evolution resulting from an action taken by the population. As an evolutionary algorithm, changes in optimal local solutions for the contemporary population effectively demonstrate this effect. Therefore, we use the percentage improvement of the current populations' optimal fitness function value compared to that of previous generations. When the population becomes worse instead of finding a better task plan, the reward is updated in the form of $penalty$. The reward reads as follows.

$$R_t = \frac{f_t^{best} - f_{t-1}^{best}}{f_{t-1}^{best}} \tag{29}$$

where f_t^{best} denotes the optimal fitness function value in the current population and f_{t-1}^{best} denotes the optimal fitness function value in the last generation population.

Higher reward values imply improved population search performance, effectively reflecting the impact of control parameters on the process of population evolution.

4. Transition

The state transition records the changes in the agent's states, actions taken, and rewards obtained. Since the policy gradient method is employed to train the network model, it becomes inconsequential whether the state transition can be recorded or not. This is because the policy gradient approach utilizes gradient descent for directly optimizing subsequent strategies toward achieving desired rewards. Although state shifts do not have an impact on strategy updates, the triple of (S_t, A_t, R_t) needs to be stored for updating the network model.

5. Policy Gradient Training Method

The training effect of the GRU network model can significantly impact the solution of the EESSP by leveraging LAGA. To optimize algorithm control parameters in continuous space, a suitable combination scheme must be identified. The policy gradient method, a typical reinforcement learning approach, is effective for optimizing model parameters through the trajectory sampling of a batch. Since the LAGA uses a population to search the solution space, the batch can be replaced by the population size.

Our study employs a policy gradient approach to update network model parameters by conducting a gradient descent on the objective function of rewards. The training process for this network is as follows:

$$\theta_{t+1} = \theta_t + \alpha \nabla_\theta L\left(\theta_t\right) \tag{30}$$

where $\nabla_\theta L\left(\theta_t\right)$ is the gradient of the reward function and α is the learning rate. $\nabla_\theta L\left(\theta_t\right)$ can be further expressed as:

$$\nabla_\theta L\left(\theta_t\right) = \mathbb{E}_{\tau\sim\theta_t}\left[\sum_{t'=0}^{\mathrm{T}} \nabla_\theta \log \pi_\theta\left(A_t \left| S_t\right.\right) \sum_{t=t'}^{\mathrm{T}} r_t\right] \tag{31}$$

Prior to training the network model, it is essential to identify the objective function of the reward and determine the trajectory. By leveraging the MDP process constructed through population evolution, a trajectory can be derived from states and actions. When the number of trajectories is reached at N_{tra}, the GRU network model parameters are updated according to the agent states, actions, and rewards using the Eq. (25)-(29). A REINFORCE Monte Carlo method is used, then the expectation of $\nabla L_\theta(\theta)$ can be approximated by sampling N_{tra} trajectories to obtain:

$$\nabla L_\theta(\theta) \approx \frac{1}{L}\sum_{i=1}^{L} r\left(\tau^i\right) \sum_{t=0}^{T-1} \nabla_\theta \ln \pi\left(A_t^{(i)} = a_t^{(i)} \mid S_t^{(i)} = s_t^{(i)}\right) \tag{32}$$

where a_i^t denotes the value of the action belonging to the ith trajectory at time step t.

Then, the network model parameters can be trained in this way. The pseudo-code of the policy gradient method is shown in Algorithm 1.

As depicted in Algorithm 1, the policy gradient training method iterates through multiple epochs for each problem scenario. To obtain control parameters (Line 9), one epoch requires the computation of state values which are derived from task scheduling results and other features. The LAGA algorithm generates new populations (Line 10) and evaluates their fitness through population evolution (Line 12). After completing a series of trajectories of the algorithm runs, the network model parameters $\mathbf{W}$ (Line 14) are updated using the backpropagation method.

Algorithm 1: Policy Gradient Training Method

Input: max epoch $Epoch$, population size N_p, the number of trajectories N_{tra}, learning rate α, max time step TS_{max}.
Output: Updated $\mathbf{W}$.

```
1  Initialize the GRU model parameters W;
2  for epoch = 1 → Epoch do
3      for tra = 1 → N_tra do
4          Set t ← 1, H_0 = 0;
5          Initialize LAGA parameters;
6          P ←Generate an initial population randomly;
7          while t ≤ TS_max do
8              Get the latest state S_t;
9              Ω_t = [CR_t, MR_t] ←Generate control
                 parameters by GRU (S_t, H_{t-1}, W);
10             P_t ←Population Evolution by
                 LAGA(P_{t-1}, Ω_t, CO, MO);
11             F_t ←Calculate the fitness function value of the
                 new population(P_t);
12             R_t ←Calculate reward using Eq. (29);
13             t ← t + 1
14     Update W using Eq. (30) and Eq. (32);
```

Algorithm training through multiple epochs enables the GRU model to acquire a range of parameter configurations that yield favorable prediction outcomes. The trained network model is then used in the LAGA presented in the subsequent section to provide the two control parameters MR_t and CR_t for the evolution of the tth generation population.

C. Learning Adaptive Genetic Algorithm

The LAGA incorporates reinforcement learning techniques into the genetic algorithm framework, enabling the algorithm to assimilate valuable information gleaned from population evolution. A trained artificial neural network model is then utilized to provide decision support for control parameters. The effective configuration of control parameters enables LAGA to find the optimal solution. During the initial stage of the search, the algorithm should focus on exploring new solution spaces, while in contrast, it is worthwhile to concentrate on a small search area when reaching a certain stage of the search. While utilizing the network model to acquire parameters, an adaptive crossover approach is employed within the genetic algorithm. This adaptive crossover technique effectively ensures the algorithm's generalization ability and facilitates the easy selection of operators that contribute to discovering superior solutions. Additionally, a population initialization method and a local search strategy have been devised to further enhance the algorithm's search performance. The quality of the initial population has

a significant impact on the performance of population search, while local search can enhance algorithmic exploitation in the local search space.

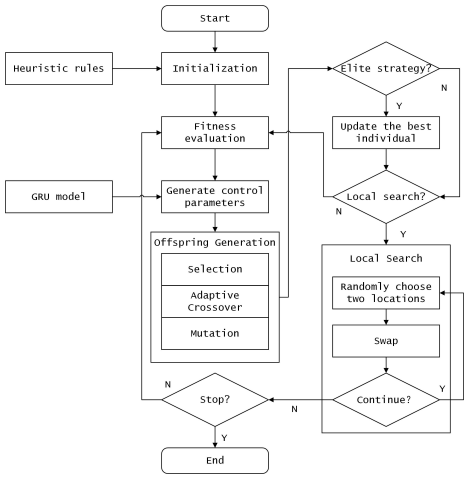


Fig. 3: The overall framework of LAGA

Figure 3 illustrates the overall framework of the LAGA. The proposed algorithm differs from the traditional genetic algorithm in several ways. Firstly, rather than being manually set by designers, control parameters are determined through a nonlinear predictive approach utilizing an artificial neural network model. Secondly, this algorithm incorporates multiple enhancement strategies including population generation and evolution techniques as well as advanced search methods that will be further explained in subsequent sections.

1. *Algorithm Overall Process*

The LAGA employs a reinforcement learning approach to train the GRU model, which can provide optimal control parameters for population evolution operators. A typical genetic algorithm framework comprises initialization, individual selection, fitness evaluation, and variation stages. Prior to each generation of population search, the LAGA utilizes a trained artificial neural network model to obtain two primary control parameters based on real-time information. The adjustment of parameters allows the algorithm to choose an appropriate search method based on environmental information. It contributes to finding the optimal solution. LAGA is coded using the integer number, with each gene representing a task. When decoding is carried out, the algorithm uses the method in [38], which arranges tasks one by one. The algorithm adopts adaptive crossover, an individual evolution operator that selects the superior operator based on search performance. The crossover operators dynamically adjust the probability of operators by evaluating iterative search. It is noteworthy that the algorithm can also adapt its search strategy according to population search performance, transitioning from exploration across the entire solution space to exploitation of local space for discovering better solutions. The pseudo-code of the LAGA is shown in Algorithm 2.

Algorithm 2: Learning Adaptive Genetic Algorithm

Input: population size N_p, GRU parameters $\mathbf{W}$, task set T, time window set TW, control parameter of elite strategy $Thre_1$, control parameter of local search $Thre_2$
Output: $Solution$

1 Initialize algorithm parameters;
2 $P \leftarrow$Generate initial population by Algorithm 3 and calculate fitness;
3 Set $t \leftarrow 1, tri_1 \leftarrow 0, tri_2 \leftarrow 0$;
4 **while** *termination criterion is not met* **do**
5 Get the latest state S_t;
6 $\Omega_t = [CR_t, MR_t], H_t \leftarrow$Generate control parameters by GRU $(S_t, H_{t-1}, \mathbf{W})$;
7 **for** $i = 1 \rightarrow N_p$ **do**
8 $indi \leftarrow$Roulette chooses individuals(P);
9 **if** $rand() < CR_t$ **then**
10 Perform adaptive crossover$(indi)$;
11 **if** $rand() < MR_t$ **then**
12 Perform mutation$(indi)$;
13 $local_best_indi, local_best \leftarrow$Evaluate the fitness function value(P);
14 Update the scores of crossover operators;
15 **if** $local\_best > gobal\_best$ **then**
16 $gobal\_best \leftarrow local\_best$;
17 $gobal\_best\_indi \leftarrow local\_best\_indi$;
18 **else**
19 $tri_1 \leftarrow tri_1 + 1$;
20 **if** $tri_1 < Thre_1$ *and* $gobal\_best! = local\_best$ **then**
21 $local\_best\_indi \leftarrow gobal\_best\_indi$;
22 **if** $local\_best < temp\_local\_best$ **then**
23 $tri_2 \leftarrow tri_2 + 1$;
24 **if** $tri_2 == Thre_2$ **then**
25 $new\_indi \leftarrow$ Local search;
26 $P \leftarrow$ Update population by replacing the worst individual with new_indi;
27 $tri_2 \leftarrow 0$;
28 $t \leftarrow t + 1$;
29 $temp\_local\_best \leftarrow local\_best$;

As expressed in Algorithm 2, the LAGA first generates a population of N_p individuals. The initial population is generated through a heuristic initialization method (Line 2). Upon completion of the fitness evaluation for the initial population, the algorithm proceeds with an iterative search for optimal solutions. In the search process, an elite strategy (Lines 20-22) and a local search method are used (Lines 25-29) in addition to the crossover and mutation operators that contain the genetic algorithm. The final detection task plan will be determined by the optimal solution obtained from the search algorithm.

2. *Initialization*

The population evolution is driven by the initial population, and a series of selection and variation operators are applied to obtain a high-quality plan. The population initialization aims to strategically position individuals within the search space while maintaining diversity. This approach significantly reduces the number of searches required to discover an optimal task plan. Therefore, an algorithm is designed with a population initialization method that combines heuristics and randomization. The

heuristic rule is donated as UPF. UPF rule is described as follows and the equation for the unit profit of the task (up_j) is calculated by $up_j = p_j/d_j$.

Heuristic rule: Compute the unit profits of tasks and generate an individual based on the descending order of their index values.

If every individual in the population follows the above heuristic rule will make the chromosome structure between individuals highly similar, which will not facilitate the search. Therefore, a parameter η is used to ensure the diversity of individuals in the population. Through the setting of this parameter, some genes within individuals are added to the chromosome randomly. The parameter η denotes the proportion of chromosomes generated according to the heuristic rule within an individual. This part of the task generates a chromosome, ordered according to the heuristic rule. The rest (1-η) proportion of genes is inserted into the existing chromosome in random positions to form a complete individual. The pseudo-code of the initialization method is shown in Algorithm 3.

Algorithm 3: Initialization

Input: population size N_p, task set T
Output: initial Population P_0
1 Set $\eta \leftarrow 0, PI = [\,]$;
2 **for** $i = 1 \rightarrow N_p$ **do**
3 $\eta \leftarrow i/(N_p + 1)$;
4 $indi_i \leftarrow$ Select tasks using UPF(T, η);
5 $PI \leftarrow$Generate a set of tasks to be inserted;
6 **while** $PI \neq \emptyset$ **do**
7 $task \leftarrow$ Random select a task from PI;
8 $indi_i \leftarrow$ Select a position randomly and insert $task$;
9 Remove the $task$ from PI;

As expressed in Algorithm 3, a certain proportion of chromosomes for each individual within the population is generated according to the heuristic rule (Line 4), while the remaining part generates a task set (Line 5) and uses a random approach to select genes (Line 7) and insert them to the chromosome. After generating the initial population, the population search will be carried out.

3. *Fitness Evaluation*

The purpose of fitness evaluation is to enable the LAGA to identify the parent individuals for the change operation from the population based on individual performance. In our EEESSP problem, our objective function aims to maximize the profit of the detection task sequence. Therefore, evaluating individual fitness follows a similar approach to calculating the objective function. The specific calculation is shown in Eq. 5. The fitness evaluation results will also facilitate the computation of RL reward values.

4. *Individual Selection*

Individual selection selects individuals from the population according to a certain strategy. Then, offspring will generate based on the selected individual. Individual selection is usually done by roulette, *k*-tournament, etc. In LAGA, a roulette wheel method is used to select individuals from the population for subsequent evolution. The equation for the roulette selection of individuals is:

$$\bar{p}_i = \frac{f_i}{\sum\limits_{i \in P} f_i} \tag{33}$$

where $\bar{p}_i$ denotes the probability value of individual i being selected, f_i denotes the fitness value.

After the selection of individuals for variation, crossover or mutation will be performed to generate offspring. Variation plays a crucial role in achieving the optimal solution. A comprehensive explanation of the variation process is provided in the subsequent section.

5. *Variation*

Variation consists of two population evolution operators: crossover and mutation. The main difference between crossover and mutation is the degree of chromosome change within an individual. The population search is constrained by parameter control to ensure efficient exploration and exploitation throughout the process, enabling the algorithm to approach or reach optimal solutions. The probability of crossover and mutation are adjusted according to specific problem scenarios. Population evolution can also be influenced by other factors, such as the length of the selected gene fragment within an individual and the type of variation. The following section describes the specific evolution operators of crossover and mutation.

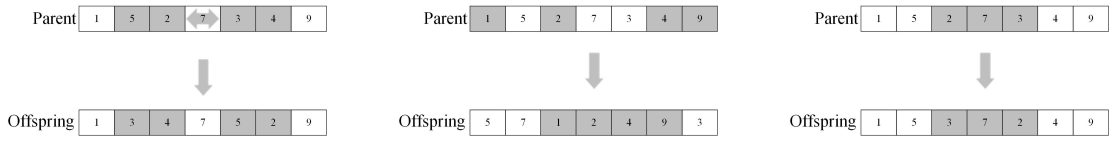
Crossover is a frequently used population evolution operator in genetic algorithms during the search process, aiming to explore the entire solution space. The offspring generated by this operator will exhibit significant differences from their parent individuals. An adaptive crossover operation has been implemented in LAGA. This adaptive crossover operator initializes each crossover rule with the same score during algorithm parameter initialization. Subsequently, weights are assigned to the rules based on their scores, and a random selection method is employed to choose a rule for crossover according to these weights. This approach bears a striking resemblance to individual selection. The equation for calculating the weights is:

$$\hat{p}_i = \frac{soc_i}{\sum\limits_{i \in R} soc_i} \tag{34}$$

where $\hat{p}_i$ denotes the probability of the ith crossover rule, soc_i denotes the score of the ith crossover rule, and R denotes the set of crossover rules.

The crossover rules used in crossover operators are divided into three types, two-point crossover rule, multi-point crossover rule, and fragment flipping crossover rule, respectively. Figure 4 gives an example of the three crossover rules.

Two-point crossover: Two gene fragments of equal length are obtained from two distinct loci within the parent genome. Without altering their internal sequence, these two fragments undergo a positional exchange to generate offspring.



(a) Two point crossover (b) Multi-point crossover (c) Fragment flipping crossover

Fig. 4: Crossover Operation

Multi-point crossover: Multiple genes are selected from the parent, and a new gene fragment is formed based on their relative order. Subsequently, the fragment is randomly inserted into the remaining portion to generate offspring.

Fragment flipping crossover: Choose the starting and ending points of a gene fragment from the parent sequence. A new fragment is then synthesized in reverse order, relative to the selected region, and inserted at the same position as its parental template to generate offspring.

These three crossover operations have varying degrees of impact on the chromosomes within individuals. However, it is challenging to determine the significance of each crossover operator. The objective is to optimize the effectiveness of each population's evolution. Therefore, upon selection and implementation of a specific crossover operation, the algorithm updates the score for that particular operator based on changes in individual fitness. The updated value's score is determined by its search performance. If the fitness value of the offspring is increasing compared to the fitness value of the parent, the score is increased by μ_1; otherwise, the score is increased by μ_2. When a certain number of times is reached, the weights of the crossover rules will be updated according to the latest score value according to Eq. (34).

Compared to crossover, mutation is a simpler process accomplished through double-point swapping. Specifically, two genes are randomly selected from a parent chromosome and their positions are exchanged to produce an offspring.

6. *Elite Strategy*

The elite strategy is specifically designed to enhance the convergence performance of the algorithm. Following the completion of each generation's population search, the most optimal individual discovered during the search process is directly incorporated into the offspring population. The inclusion of an elite individual effectively accelerates convergence speed, enabling rapid discovery of higher-quality task plans during population searches. When the population search reaches the threshold $Thre_1$, the elite strategy is no longer used.

7. *Local Search*

Local search (LS) is a method employed to identify the local optimum within a specific solution space, which can often play a pivotal role when the search proves ineffective across the entire solution space. A 2-opt local search operation is utilized as a simple and efficient approach to update the neighborhood structure. In this 2-opt procedure, two genes are randomly selected from the best individual discovered during population search thus far, and their positions are exchanged to generate a new individual. A fitness evaluation and comparison process will determine whether to continue with the local search, which must be effectively balanced with the global search. If too many searches are conducted, the algorithm solution may become trapped in local optimization without escaping. Therefore, when there is no further improvement in individual fitness value, the local search stage should end and the algorithm should return to the population search stage. The pseudo-code of the local search is shown in Algorithm 4.

Algorithm 4: Local Search

Input: gobal best individual $gobal\_best\_indi$, fitness of gobal best individual $gobal\_best$
Output: new individual new_indi

```
1 while termination criterion is not met do
2     gene_1, gene_2 ← Random select two genes from gobal_best_indi;
3     new_indi ← Swap two gene positions to generate a new individual;
4     fitness ← Calculate the fitness function value (new_indi);
5     if fitness > gobal_best then
6         gobal_best ← fitness;
7     else
8         Loop While;
```

As expressed in algorithm 4, two genes are randomly selected from the chromosome of the best individual in the population search (Line 2), and the positions are exchanged to obtain offspring (Line 3). After that, the value of the fitness function is evaluated to determine whether to continue the local search process or to return to the population search (Lines 5-8). The local search algorithm will update the best individual for the population search at the end of the algorithm.

8. *Complexity Analysis*

In the LAGA, the complexity of the GRU model is $O(Batch * |T|^2 * d)$, where d denotes the number of features. The complexity of the GA algorithm population search generation is $O(Batch * |T| * |TW|)$, and the complexity of the local search is $O(|T| * |TW|)$. Since $|TW| >> d$. So the time complexity of the LAGA is

$O(Batch * |T| * |TW|)$. While batch uses population P instead, the time complexity can also be rewritten as $O(|P| * |T| * |TW|)$.

V. Experiment

A. Experimental Setup

Experimental environment: The experiments in this study are done on a desktop computer with Intel(R) Core(TM) I7-7700 3.6 GHz CPU, 16 GB RAM, and NVIDIA GeForce 2070Ti. The coding environment is Python 3.9.7.

Comparison algorithms: The problem-solving performance of the proposed algorithm is verified using four algorithms, which consist of three evolutionary algorithms and an individual search algorithm based on neighborhood structure improvement. The population perturbation and elimination strategy based genetic algorithm (GA-PE) [23], improved adaptive large neighborhood search algorithm with tubu search strategy (ALNS-I) [42], artificial bee colony algorithm (ABC) [43], and improved ant colony algorithm with adaptive assignment strategy (IACO) [44] were selected. These algorithms have been successfully applied to solve EOSSP problems. The parameters of all algorithms are shown in Table I. Due to the limited space constraints of the paper, the details of the comparison algorithms are provided in the supplementary material.

Experimental scenarios: Scenarios with different task scales are used to evaluate the scheduling performance of the algorithm in all aspects. Since there is no public benchmark for the EESSP, we use a random approach to generate a series of scenarios. Specifically, the entire scheduling period is 24 hours. The earliest allowed detection time and the latest allowed completion time for all tasks fall within this range. The duration of each detection task obeys a normal distribution with a mean of 70s and a standard deviation of 50s. The profit obeys a uniform distribution with a mean of 12.5 and a standard deviation of 7.5. Tasks are chosen randomly on a global scale. One of the satellite orbit parameters is shown in Table II. The maximum allowable detection angle of the satellite is 80 degrees, the detection mode conversion time is 10s, the bandwidth conversion time is 15s, and the frequency band conversion time is 20s. The source code of LAGA and data will be released to a GitHub repository [1].

To succinctly represent the scenario, the "A-B" format is employed to depict a given scenario. In this context, "A" signifies the number of tasks while "B" denotes the specific scenario for task-scale "A". During the experiment, task sizes range from 100 to 1000. For the convenience of differentiation, the scenarios are artificially divided into small-scale sets (denoted as Set I), medium-scale sets (denoted as Set II), and large-scale sets (denoted as Set III).

[1]https://github.com/tomsong00/LAGA

Evaluation metrics: All algorithms run 30 times in each scenario to evaluate the algorithms' performance. The optimal profit (denoted as Best), the mean profit (denoted as Mean), and the standard deviation (denoted as Std.) are set as evaluation indicators. Based on the data obtained, the Wilcoxon rank-sum test (denoted as WR) is used to analyze whether the differences between the results obtained by different algorithms are significant [45]. In terms of the algorithm convergence performance, it is evaluated through convergence curves. Furthermore, the effect of the strategy used and parameter sensitivity is analyzed.

B. Results and Analysis

Firstly, the performance of the algorithms in Set I is verified. As shown in Table III, finding the optimal solution in a 100 task-scale scenario is not difficult for all algorithms used in the experiments. Moreover, the majority of the algorithms' search performance is stable, and only ALNS-I still has some gap between the average and optimal values of the profits obtained in scenario 100-4. Starting from scenarios with a 200-task scale, the proposed algorithm can find better optimal values than the compared algorithms. Among these comparison algorithms, IACO can obtain good scheduling performance while maintaining high stability. In addition, the WD indicator shows that the LAGA differs significantly from the other algorithms in most of the scenarios.

The LAGA performs as well in Set II as in Set I. As shown in Table IV, it is evident that the optimal performance is achieved in most scenarios, except for a few instances where stability may be compromised. Furthermore, there is a significant increase in the disparity between algorithms compared to the results obtained in Set I. In the 600 task-scale scenarios, the difference between LAGA and other comparison algorithms' optimal and average profits can reach approximately two hundred.

After using Set II to validate algorithms, the task scale is further increased and the results in Set III are shown in Table V. The experiments on the algorithm's solving ability in large-scale scenarios can effectively demonstrate the balanced performance of the algorithm's exploration and exploitation, as well as its application prospects. It is apparent that the LAGA still demonstrates exceptional performance in Set III owing to its effective amalgamation of population exploration and local space exploitation, which can be highly advantageous for enhancing solution quality given the vast solution space.

To observe the solution performance of the algorithm more intuitively, the results of the average performance of the 300 task-scale scenarios are presented in the form of a bar chart. As shown in Fig. 5(a)-(f), the average profit obtained by the proposed algorithm is significantly higher than the other three algorithms. And the search performance of the other three algorithms is close. From the paired statistical tests, the results obtained by LAGA are

TABLE I: Parameters of algorithms

LAGA	GA-PE [23]	ALNS-I [42]	ABC [43]	IACO [44]
maximum fitness evaluation times 5000 population size 10 number of trajectories 10 learning rate 0.001 initial score 50, update score 30, 10 threshold 2000, 20	maximum fitness evaluation times 5000 population size 10 crossover probability 0.9 mutation probability 0.05 population perturbation threshold 10	maximum fitness evaluation times 5000 maximum iteration of no improvement 1000 percent of tasks to remove 10% weight update parameter 0.5 coefficient of annealing 0.9975 score increment 110, 20, 30	maximum fitness evaluation times 5000 population size 10 minimum population size of search population 3 scout bee size 1 control parameter limit 20 the degree of fitness acceptance 0.9	maximum fitness evaluation times 5000 maximum iterations for Single assignment 100 number of ants 15 influence parameter uniform [20,40] evaporation rate 0.5 initial pheromone 100, importance 1, profit importance 2

TABLE II: Orbital parameters of one satellite

Attributes	Values
the length of semi-major axis	7000km
eccentricity	0.00015
inclination	97.672°
argument of perigee	0°
the right ascension of the ascending node	21.75°
mean anomaly	158.25°

significantly different from the state-of-the-art algorithms at the level of $p < 0.001$.

In addition to statistical analysis of algorithm differences, convergence performance is another significant evaluation criterion for search algorithms. Convergence curves for 800 and 1000 task-scale scenarios are analyzed. As shown in Fig. 6. All these five algorithms have good global search capability and can exploit solutions in the local space. The heuristic population initialization method used in the LAGA algorithm has a positive effect on finding the optimal solution, and this initialization method has outstanding performance in 1000 task-scale scenarios. After the search process starts, the LAGA has a fast convergence speed and can start a new search through the search strategy after the convergence encounters a bottleneck.

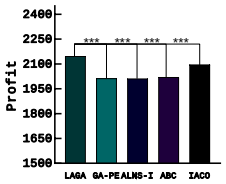
Then, a variation LAGA that uses a random way to generate the initial population (denoted as LAGA/RI) is used to compare with LAGA. The results for 300, 600, 800, and 1000 task-scale scenarios are shown in Figure 7(a)-(d). From the results, it can be seen that the use of the heuristic initialization method can effectively improve the search performance of the LAGA. This initialization method can effectively utilize the knowledge and improve the effectiveness of the algorithm in finding solutions while maintaining the diversity of populations. It is clear that as the task-scale increases, the role of the heuristic population initialization method tends to diminish and then increase. This is because the complexity of the problem depends more on the search process and solutions obtained by initialization are not decentralized throughout the solution space. Then, when the search space is large enough, knowledge again drives the population search in a good direction. In summary, the above experimental results show that the LAGA performs significantly better than other state-of-the-art algorithms in solving the EESSP. From several metric aspects, the GA framework combined with an artificial neural network model and various strategies designed is effective.

For parameter $Thre_1$, the setting of the value affects the scheduling performance of LAGA. Figure 8 shows the results of sensitivity analysis for two scenarios with different parameter settings of 1000, 1500, and 2000 at 800-1 and 800-2. For scenario 800-1, the average performance of the algorithm improves as $Thre_1$ increases. The parameter setting of 2000 is the most reasonable. Similarly, $Thre_1$ is set to 2000 can help the algorithm to obtain good planning performance with little volatility in scenario 800-2. Therefore, it is reasonable to set $Thre_1$ to 2000.

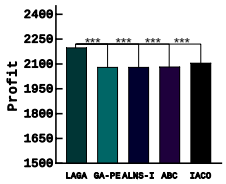
C. Case Study

We use a real-world application case to provide a clearer understanding of the electromagnetic detection scenario and the algorithm planning process (see Figure 9). In a satellite system containing a total of 20 EESs, users submit a large number of requirements to the satellite management system every day. The system is pre-processed and generates many tasks to be planned. In general, the number of tasks per day is not equal. Here, Table VI gives the number of tasks per day. Our target is to use the algorithm to plan the electromagnetic detection schedule for one week. The attributes of the tasks are kept consistent with those in the experimental setup. Figure 10 shows the planning performance of the LAGA algorithm and the heuristic algorithm (HA). The detection plan is generated by HA through ranking tasks based on their earliest allowable detection time, which is also the most commonly utilized algorithm in actual satellite systems.

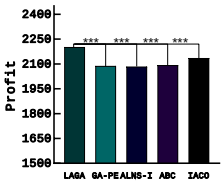
As can be seen from Figure 10, the LAGA algorithm planning performs better than HA. The iterative search enables LAGA to efficiently identify an optimal combination of detection tasks from a vast pool of options. Typically, managers make minor adjustments to the scenario based on practical considerations. Once the final solution is formed, the satellite management system determines upload and downlink plans based on ground station availability. Subsequently, the system generates commands based on each command template. Once the designated time is reached, the satellite will execute the mission and generate data. Subsequently, these data are transmitted to the ground for distribution among respective users.



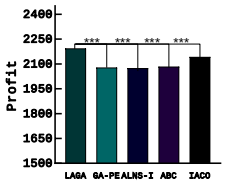
(a) Average performance of scenario 300-1



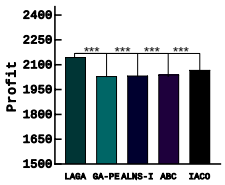
(b) Average performance of scenario 300-2



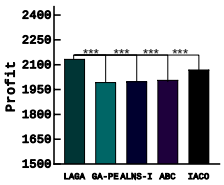
(c) Average performance of scenario 300-3



(d) Average performance of scenario 300-4

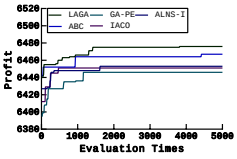


(e) Average performance of scenario 300-5

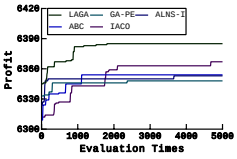


(f) Average performance of scenario 300-6

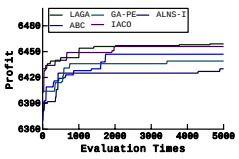
Fig. 5: Average performance in different scenarios



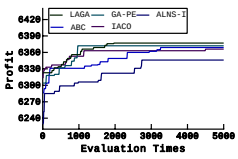
(a) Convergence curves of scenario 800-1



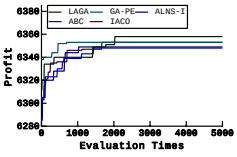
(b) Convergence curves of scenario 800-2



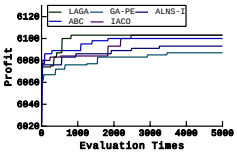
(c) Convergence curves of scenario 800-3



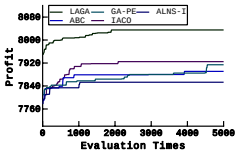
(d) Convergence curves of scenario 800-4



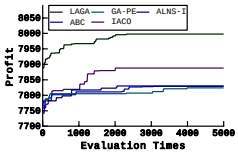
(e) Convergence curves of scenario 800-5



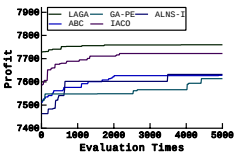
(f) Convergence curves of scenario 800-6



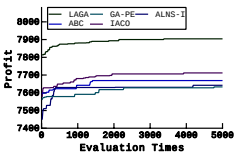
(g) Convergence curves of scenario 1000-1



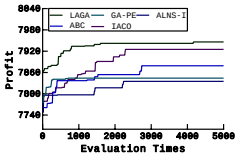
(h) Convergence curves of scenario 1000-2



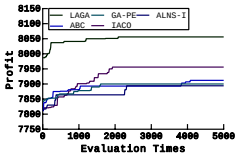
(i) Convergence curves of scenario 1000-3



(j) Convergence curves of scenario 1000-4

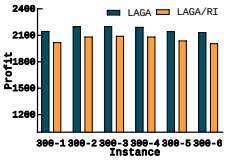


(k) Convergence curves of scenario 1000-5

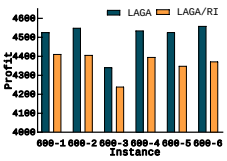


(l) Convergence curves of scenario 1000-6

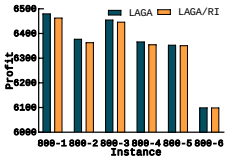
Fig. 6: Convergence curves of 800 and 1000 task-scale scenarios



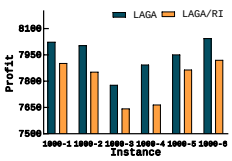
(a) Results under 300 task-scale scenarios



(b) Results under 600 task-scale scenarios



(c) Results under 800 task-scale scenarios



(d) Results under 1000 task-scale scenarios

Fig. 7: Results of different population initialization methods used in different scenarios

TABLE III: Scheduling Results for Set I

Instance	LAGA			GA-PE [23]				ALNS-I [42]				ABC [43]				IACO [44]			
	Best	Mean	Std.	Best	Mean	Std.	WR	Best	Mean	Std.	WR	Best	Mean	Std.	WR	Best	Mean	Std.	WR
100-1	**891**	**891.00**	**0.00**	**891**	**891.00**	**0.00**	=	**891**	**891.00**	**0.00**	=	**891**	**891.00**	**0.00**	=	**891**	**891.00**	**0.00**	=
100-2	**805**	**805.00**	**0.00**	**805**	**805.00**	**0.00**	=	**805**	**805.00**	**0.00**	=	**805**	**805.00**	**0.00**	=	**805**	**805.00**	**0.00**	=
100-3	**806**	**806.00**	**0.00**	**806**	**806.00**	**0.00**	=	**806**	**806.00**	**0.00**	=	**806**	**806.00**	**0.00**	=	**806**	**806.00**	**0.00**	=
100-4	**819**	**819.00**	**0.00**	**819**	**819.00**	**0.00**	=	**819**	817.80	2.73	-	**819**	**819.00**	**0.00**	=	**819**	**819.00**	**0.00**	=
100-5	**820**	**820.00**	**0.00**	**820**	**820.00**	**0.00**	=	**820**	**820.00**	**0.00**	=	**820**	**820.00**	**0.00**	=	**820**	**820.00**	**0.00**	=
100-6	**822**	**822.00**	**0.00**	**822**	**822.00**	**0.00**	=	**822**	**822.00**	**0.00**	=	**822**	**822.00**	**0.00**	=	**822**	**822.00**	**0.00**	=
200-1	**1550**	**1542.70**	**4.65**	1544	1530.63	5.57	-	1539	1529.47	5.66	-	1544	1534.87	4.71	-	1549	1540.19	4.73	=
200-2	**1666**	1648.03	7.91	1628	1605.33	8.61	-	1621	1604.53	9.01	-	1624	1607.90	**6.34**	-	1662	**1651.68**	8.14	+
200-3	**1615**	**1608.27**	**3.49**	1601	1586.57	5.63	-	1596	1585.07	5.54	-	1606	1588.47	6.05	-	1602	1589.26	5.28	-
200-4	**1677**	**1671.57**	**4.55**	1672	1662.90	4.82	-	1675	1660.50	5.61	-	1680	1664.47	5.08	-	1670	1659.81	4.67	-
200-5	**1535**	**1523.70**	**4.33**	1526	1516.13	4.80	-	1530	1516.33	5.65	-	1528	1519.50	5.60	-	1529	1520.33	4.42	-
200-6	**1589**	**1581.93**	**5.35**	1570	1553.17	7.21	-	1566	1553.13	5.92	-	1573	1558.23	6.41	-	1567	1561.58	5.73	-
300-1	**2166**	**2147.47**	**8.56**	2041	2013.50	11.42	-	2037	2012.10	10.66	-	2065	2020.80	12.34	-	2109	2096.13	9.75	-
300-2	**2216**	**2199.67**	**8.09**	2110	2081.83	11.02	-	2106	2081.17	11.14	-	2104	2084.23	8.70	-	2118	2106.94	8.33	-
300-3	**2211**	**2202.20**	**4.21**	2107	2088.00	9.31	-	2114	2083.97	9.40	-	2124	2092.37	8.82	-	2145	2135.12	7.16	-
300-4	**2205**	**2194.03**	**7.42**	2094	2078.30	8.30	-	2100	2074.63	11.63	-	2105	2084.10	7.78	-	2152	2142.47	7.74	-
300-5	**2161**	**2145.63**	**7.70**	2051	2030.83	7.51	-	2057	2033.80	9.83	-	2066	2040.23	7.90	-	2077	2067.65	7.87	-
300-6	**2153**	**2134.20**	**10.46**	2035	1994.63	15.91	-	2049	1999.20	16.41	-	2030	2007.77	10.55	-	2083	2070.09	10.92	-
	+/-/=			0/12/6				0/13/5				0/12/6				1/7/10			

TABLE IV: Scheduling Results for Set II

Instance	LAGA			GA-PE [23]				ALNS-I [42]				ABC [43]				IACO [44]			
	Best	Mean	Std.	Best	Mean	Std.	WR	Best	Mean	Std.	WR	Best	Mean	Std.	WR	Best	Mean	Std.	WR
400-1	**3096**	**3092.00**	**2.78**	**3096**	3087.20	4.80	-	**3096**	3088.10	3.96	-	**3096**	3090.40	3.67	-	3096	3091.43	3.35	=
400-2	**3252**	**3238.97**	**4.53**	3236	3223.27	5.87	-	3237	3223.67	5.97	-	3239	3230.07	6.55	-	3244	3237.06	4.68	=
400-3	**3305**	**3290.40**	**7.06**	3284	3257.40	10.24	-	3270	3254.60	9.43	-	3284	3266.43	8.36	-	3287	3261.18	8.19	-
400-4	**3097**	**3096.37**	**1.50**	3097	3089.03	5.62	-	3097	3087.17	5.47	-	3097	3092.17	3.91	-	3097	3073.95	4.1	-
400-5	**3059**	**3049.30**	**5.05**	3058	3040.43	7.21	-	3051	3038.40	6.12	-	3055	3044.87	5.31	-	3056	3041.11	5.05	-
400-6	**3115**	**3105.63**	**4.23**	3107	3091.27	5.26	-	3107	3090.60	6.25	-	3103	3095.97	4.58	-	3108	3094.92	4.47	-
500-1	**4061**	**4037.67**	**10.01**	3946	3915.50	15.43	-	3946	3913.00	12.76	-	3965	3932.80	13.16	-	3982	3974.03	12.92	-
500-2	**3735**	**3724.10**	**7.57**	3650	3618.37	13.56	-	3641	3620.90	9.42	-	3673	3635.07	13.15	-	3690	3672.84	8.09	-
500-3	**3894**	**3876.57**	**8.02**	3774	3745.93	12.38	-	3768	3743.57	10.29	-	3785	3758.10	11.96	-	3821	3800.92	10.92	-
500-4	**3987**	**3968.80**	**9.67**	3924	3890.30	12.26	-	3923	3888.10	14.09	-	3922	3902.20	9.41	-	3936	3917.15	10.08	-
500-5	**3831**	**3810.67**	**8.06**	3740	3694.00	14.10	-	3718	3688.20	13.67	-	3740	3702.13	13.87	-	3779	3755.93	10.72	-
500-6	**3896**	**3881.77**	**8.48**	3841	3815.93	11.00	-	3835	3816.63	8.93	-	3850	3828.50	10.33	-	3852	3836.29	8.86	-
600-1	**4541**	**4526.23**	**12.93**	4399	4339.70	20.39	-	4366	4334.43	15.22	-	4380	4326.23	13.61	-	4423	4341.02	13.52	-
600-2	**4577**	**4549.37**	**10.51**	4344	4316.00	13.55	-	4386	4318.17	18.61	-	4374	4334.03	13.97	-	4482	4469.12	11.17	-
600-3	**4361**	**4341.60**	**9.89**	4197	4164.73	14.68	-	4197	4161.73	13.67	-	4205	4180.00	12.98	-	4351	4328.37	10.25	-
600-4	**4575**	**4535.70**	14.11	4347	4308.70	19.11	-	4335	4310.67	13.23	-	4354	4329.97	**13.19**	-	4482	4462.6	13.65	-
600-5	**4553**	**4526.77**	**12.48**	4325	4256.63	19.46	-	4313	4265.70	18.49	-	4302	4274.57	14.75	-	4419	4392.17	13.92	-
600-6	**4580**	**4559.20**	**10.11**	4350	4296.10	19.45	-	4342	4294.30	22.75	-	4347	4315.77	15.64	-	4406	4384.53	14.75	-
	+/-/=			0/18/0				0/18/0				0/18/0				0/16/2			

TABLE V: Scheduling Results for Set III

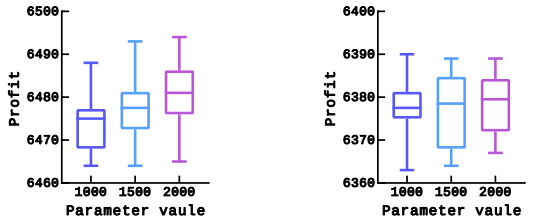
Instance	LAGA			GA-PE [23]				ALNS-I [42]				ABC [43]				IACO [44]			
	Best	Mean	Std.	Best	Mean	Std.	WR	Best	Mean	Std.	WR	Best	Mean	Std.	WR	Best	Mean	Std.	WR
800-1	**6494**	**6481.20**	6.57	6480	6457.77	9.26	-	6477	6455.57	10.74	-	6475	6463.80	**6.19**	-	6481	6471.64	6.28	-
800-2	**6389**	**6378.27**	6.59	6371	6351.33	7.15	-	6375	6355.70	9.71	-	6379	6364.03	7.43	-	6380	6368.17	**6.54**	-
800-3	**6462**	**6455.97**	**4.36**	6449	6438.57	6.95	-	**6462**	6441.17	8.27	-	**6462**	6447.17	5.34	-	**6462**	6447.22	5.62	-
800-4	**6379**	**6367.43**	**7.06**	6374	6349.20	10.33	-	6360	6348.07	7.65	-	6374	6355.57	8.87	-	6379	6356.81	7.83	-
800-5	**6361**	**6353.77**	4.35	**6361**	6347.33	5.57	-	6358	6347.43	6.22	-	6358	6351.77	**4.12**	-	**6361**	6352.81	4.38	-
800-6	**6103**	**6100.03**	3.22	**6103**	6095.03	4.44	-	**6103**	6095.67	4.33	-	**6103**	6099.80	**2.76**	-	**6103**	6097.52	4.01	-
900-1	**7018**	**7007.47**	**6.57**	6980	6963.53	10.37	-	6997	6967.73	12.33	-	6991	6976.67	7.49	-	7002	6981.62	7.16	-
900-2	**7303**	**7287.40**	**7.15**	7253	7217.40	10.89	-	7246	7216.30	13.14	-	7246	7230.23	8.23	-	7268	7243.04	7.93	-
900-3	**7006**	**6992.50**	**10.36**	6942	6920.07	13.52	-	6964	6915.23	15.90	-	6955	6936.97	10.33	-	6963	6942.62	11.42	-
900-4	**7070**	**7051.10**	**8.88**	7013	6984.80	14.08	-	7020	6983.87	15.60	-	7016	6999.60	9.77	-	7024	7006.82	10.01	-
900-5	**7352**	**7339.97**	**7.08**	7330	7299.20	14.40	-	7341	7304.10	12.27	-	7335	7317.00	8.27	-	7342	7318.06	8.95	-
900-6	**7193**	**7175.77**	**8.24**	7154	7121.97	11.29	-	7164	7130.27	17.50	-	7169	7145.23	11.16	-	7170	7126.82	10.60	-
1000-1	**8047**	**8024.27**	**12.15**	7918	7883.97	15.21	-	7925	7880.90	18.35	-	7929	7902.97	12.46	-	7936	7902.15	14.92	-
1000-2	**8027**	**8004.53**	**9.17**	7878	7830.87	21.51	-	7881	7835.27	19.63	-	7885	7853.40	12.99	-	7912	7854.92	11.52	-
1000-3	**7803**	**7778.63**	**11.96**	7664	7618.47	18.07	-	7676	7622.53	19.42	-	7683	7643.80	17.35	-	7743	7628.67	13.53	-
1000-4	**7926**	**7894.03**	**11.60**	7693	7647.40	14.92	-	7683	7640.23	20.93	-	7711	7665.87	19.66	-	7842	7815.63	12.67	-
1000-5	**7976**	**7952.07**	**12.44**	7868	7846.97	12.29	-	7886	7849.33	15.74	-	7894	7865.60	13.77	-	7901	7874.29	13.61	-
1000-6	**8061**	**8044.77**	**9.07**	7925	7898.33	14.10	-	7945	7898.20	17.13	-	7972	7920.70	18.18	-	7984	7960.95	14.08	-
	+/-/=			0/18/0				0/18/0				0/18/0				0/18/0			

TABLE VI: Number of tasks in one week

Number of day	1	2	3	4	5	6	7
Task size	200	400	600	800	1000	1000	400

D. Discussion

LAGA demonstrates excellent scheduling performance in solving the EESSP problem, particularly for large-scale instances. This highlights the effectiveness of parameter control methods and other optimization strategies in enhancing GA's exploration and exploitation capabilities. The combination of global and local search enables comprehensive solution space exploration while focusing on regions that are likely to generate optimal solutions. The algorithm improvement strategies are primarily designed to address large-scale problem scenarios and can be flexibly applied to simpler problem scenarios. This implies that LAGA possesses a strong capability for practical engineering applications. Moreover, this design



(a) Results with different parameters of scenario 800-1

(b) Results with different parameters of scenario 800-2

Fig. 8: Parameter sensitivity analysis of $Thre_1$

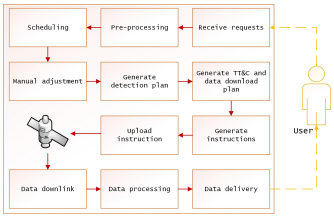


Fig. 9: Process of electromagnetic detection

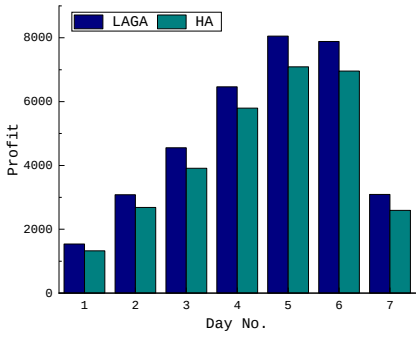


Fig. 10: One-week planning performance statistics

concept is not limited to the GA framework but can also be extended to other meta-heuristic algorithms.

VI. Conclusion

For the Earth electromagnetic satellite scheduling problem (EESSP), we formulate a mixed-integer programming model and propose a learning adaptive genetic algorithm. The LAGA effectively integrates the respective strengths of evolutionary algorithms and artificial neural networks. Leveraging the characteristics of population optimization, we employ a policy gradient-based reinforcement learning training method to train the GRU model. The algorithm in this paper also employs a range of enhancement strategies, in addition to artificial neural networks, to enhance the efficiency of the algorithm's search process. The utilization of an elite strategy enables the population searches to exhibit superior convergence performance from the outset. Furthermore, a local search method has been devised to focus on global exploration while simultaneously identifying optimal local solutions. The enhancement of the algorithm effectively achieves a balance between exploration and exploitation in the population search, facilitating the discovery of a satisfactory satellite detection plan. Extensive experiments have demonstrated the efficacy of adaptive learning methods, enabling comprehensive utilization of information acquired from the search. Correspondingly, it can allow the LAGA to obtain better plans. Compared to existing algorithms, LAGA can bring a 2%-3% improvement in profit.

In the future, we will explore alternative reinforcement learning methods and ensemble strategies, such as employing reinforcement learning techniques to determine suitable operators for population evolution. Additionally, reinforcement learning can be utilized to generate offspring populations by selecting individuals from the parent population. In the design phase of EESSP, it is imperative to consider more intricate scenarios such as uncertain environmental factors or potential equipment emergencies.

Acknowledgment

Thanks to Jie Chun, Prof. Yingwu Chen, and the reviewers for their valuable comments. Special thanks to Prof. P. N. Suganthn, Prof. Witold Pedrycz, and Dr. Yue Zhang for their help in revising the paper. Thanks to every author for his efforts. We are also very grateful to the chief editor and the associate editor for their recognition of our work.

REFERENCES

[1] X. Shen, X. Zhang, S. Yuan, L. Wang, J. Cao, J. Huang, X. Zhu, P. Piergiorgio, and J. Dai, "The state-of-the-art of the china seismo-electromagnetic satellite mission," *Science China Technological Sciences*, vol. 61, pp. 634–642, 2018.

[2] O. Kodheli, E. Lagunas, N. Maturo, S. K. Sharma, B. Shankar, J. F. M. Montoya, J. C. M. Duncan, D. Spano, S. Chatzinotas, S. Kisseleff *et al.*, "Satellite communications in the new space era: A survey and future challenges," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 1, pp. 70–109, 2020.

[3] M. Parrot, "Use of satellites to detect seismo-electromagnetic effects," *Advances in Space Research*, vol. 15, no. 11, pp. 27–35, 1995.

[4] X. Wang, G. Song, R. Leus, and C. Han, "Robust earth observation satellite scheduling with uncertainty of cloud coverage," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 56, no. 3, pp. 2450–2461, 2019.

[5] G. Wu, X. Mao, Y. Chen, X. Wang, W. Liao, and W. Pedrycz, "Coordinated scheduling of air and space observation resources via divide and conquer framework and iterative optimization," *IEEE Transactions on Aerospace and Electronic Systems*, pp. 1–12, 2022.

[6] P. Monmousseau, "Scheduling of a constellation of satellites: creating a mixed-integer linear model," *Journal of Optimization Theory and Applications*, vol. 191, no. 2-3, pp. 846–873, 2021.

[7] K. Wu, D. Zhang, Z. Chen, J. Chen, and X. Shao, "Multi-type multi-objective imaging scheduling method based on improved

nsga-iii for satellite formation system," *Advances in Space Research*, vol. 63, no. 8, pp. 2551–2565, 2019.

[8] X. Wang, G. Wu, L. Xing, and W. Pedrycz, "Agile earth observation satellite scheduling over 20 years: Formulations, methods, and future directions," *IEEE Systems Journal*, vol. 15, no. 3, pp. 3881–3892, 2020.

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

[10] H. Wang, M. Xu, R. Wang, and Y. Li, "Scheduling earth observing satellites with hybrid ant colony optimization algorithm," in *2009 International Conference on Artificial Intelligence and Computational Intelligence*, vol. 2. IEEE, 2009, pp. 245–249.

[11] L. Wei, L. Xing, Q. Wan, Y. Song, and Y. Chen, "A multi-objective memetic approach for time-dependent agile earth observation satellite scheduling problem," *Computers & Industrial Engineering*, vol. 159, p. 107530, 2021.

[12] J. Wu, F. Yao, Y. Song, L. He, F. Lu, Y. Du, J. Yan, Y. Chen, L. Xing, and J. Ou, "Frequent pattern-based parallel search approach for time-dependent agile earth observation satellite scheduling," *Information sciences*, vol. 636, 2023.

[13] Y. Song, J. Ou, J. Wu, Y. Wu, L. Xing, and Y. Chen, "A cluster-based genetic optimization method for satellite range scheduling system," *Swarm and evolutionary computation*, vol. 79, 2023.

[14] L. Haldurai, T. Madhubala, and R. Rajalakshmi, "A study on genetic algorithm and its applications," *Int. J. Comput. Sci. Eng*, vol. 4, no. 10, pp. 139–143, 2016.

[15] X. Chen, G. Reinelt, G. Dai, and A. Spitz, "A mixed integer linear programming model for multi-satellite scheduling," *European Journal of Operational Research*, vol. 275, no. 2, pp. 694–707, 2019.

[16] W. Zhu, X. Hu, W. Xia, and P. Jin, "A two-phase genetic annealing method for integrated earth observation satellite scheduling problems," *Soft Computing*, vol. 23, pp. 181–196, 2019.

[17] C. G. Valicka, D. Garcia, A. Staid, J. Watson, G. Hackebeil, S. Rathinam, and L. Ntaimo, "Mixed-integer programming models for optimal constellation scheduling given cloud cover uncertainty," *European Journal of Operational Research*, vol. 275, no. 2, pp. 431–445, 2019.

[18] J. Berger, N. Lo, and M. Barkaoui, "Quest–a new quadratic decision model for the multi-satellite scheduling problem," *Computers & Operations Research*, vol. 115, p. 104822, 2020.

[19] H. Fan, Z. Yang, X. Zhang, S. Wu, J. Long, and L. Liu, "A novel multi-satellite and multi-task scheduling method based on task network graph aggregation," *Expert Systems with Applications*, vol. 205, p. 117565, 2022.

[20] A. A. Rahmani Hosseinabadi, J. Vahidi, B. Saemi, A. K. Sangaiah, and M. Elhoseny, "Extended genetic algorithm for solving open-shop scheduling problem," *Soft computing*, vol. 23, pp. 5099–5116, 2019.

[21] S. Karakatič, "Optimizing nonlinear charging times of electric vehicle routing with genetic algorithm," *Expert Systems with Applications*, vol. 164, p. 114039, 2021.

[22] M. Barkaoui and J. Berger, "A new hybrid genetic algorithm for the collection scheduling problem for a satellite constellation," *Journal of the Operational Research Society*, vol. 71, no. 9, pp. 1390–1410, 2020.

[23] M. Chen, J. Wen, Y. Song, L. Xing, and Y. Chen, "A population perturbation and elimination strategy based genetic algorithm for multi-satellite tt&c scheduling problem," *Swarm and Evolutionary Computation*, vol. 65, p. 100912, 2021.

[24] Y. Li, R. Wang, Y. Liu, and M. Xu, "Satellite range scheduling with the priority constraint: An improved genetic algorithm using a station id encoding method," *Chinese Journal of Aeronautics*, vol. 28, no. 3, pp. 789–803, 2015.

[25] Y. Huang, Z. Mu, S. Wu, B. Cui, and Y. Duan, "Revising the observation satellite scheduling problem based on deep reinforcement learning," *Remote Sensing*, vol. 13, no. 12, p. 2377, 2021.

[26] Y. He, G. Wu, Y. Chen, and W. Pedrycz, "A two-stage framework and reinforcement learning-based optimization algorithms for complex scheduling problems," *arXiv preprint arXiv:2103.05847*, 2021.

[27] J. T. Lam, F. Rivest, and J. Berger, "Deep reinforcement learning for multi-satellite collection scheduling," in *Theory and Practice of Natural Computing: 8th International Conference, TPNC 2019, Kingston, ON, Canada, December 9–11, 2019, Proceedings 8*. Springer, 2019, pp. 184–196.

[28] L. Ren, X. Ning, and Z. Wang, "A competitive markov decision process model and a recursive reinforcement-learning algorithm for fairness scheduling of agile satellites," *Computers & Industrial Engineering*, vol. 169, p. 108242, 2022.

[29] R. Qi, J. Li, J. Wang, H. Jin, and Y. Han, "Qmoea: A q-learning-based multiobjective evolutionary algorithm for solving time-dependent green vehicle routing problems with time windows," *Information Sciences*, vol. 608, pp. 178–201, 2022.

[30] Z. Sun, U. Benlic, M. Li, and Q. Wu, "Reinforcement learning based tabu search for the minimum load coloring problem," *Computers & Operations Research*, vol. 143, p. 105745, 2022.

[31] G. Wu, R. Mallipeddi, and P. N. Suganthan, "Ensemble strategies for population-based optimization algorithms–a survey," *Swarm and evolutionary computation*, vol. 44, pp. 695–711, 2019.

[32] J. Sun, X. Liu, T. Bäck, and Z. Xu, "Learning adaptive differential evolution algorithm from optimization experiences by policy gradient," *IEEE Transactions on Evolutionary Computation*, vol. 25, no. 4, pp. 666–680, 2021.

[33] Y. Tian, X. Li, H. Ma, X. Zhang, K. C. Tan, and Y. Jin, "Deep reinforcement learning based adaptive operator selection for evolutionary multi-objective optimization," *IEEE Transactions on Emerging Topics in Computational Intelligence*, 2022.

[34] Y. Du, J. Li, X. Chen, P. Duan, and Q. Pan, "Knowledge-based reinforcement learning and estimation of distribution algorithm for flexible job shop scheduling problem," *IEEE Transactions on Emerging Topics in Computational Intelligence*, 2022.

[35] L. Wang, Z. Pan, and J. Wang, "A review of reinforcement learning based intelligent optimization for manufacturing scheduling," *Complex System Modeling and Simulation*, vol. 1, no. 4, pp. 257–270, 2021.

[36] Z. Liao and S. Li, "Solving nonlinear equations systems with an enhanced reinforcement learning based differential evolution," *Complex System Modeling and Simulation*, vol. 2, no. 1, pp. 78–95, 2022.

[37] H. Xia, C. Li, S. Zeng, Q. Tan, J. Wang, and S. Yang, "A reinforcement-learning-based evolutionary algorithm using solution space clustering for multimodal optimization problems," in *2021 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2021, pp. 1938–1945.

[38] Y. Song, L. Wei, Q. Yang, J. Wu, L. Xing, and Y. Chen, "Rl-ga: A reinforcement learning-based genetic algorithm for electromagnetic detection satellite scheduling problem," *Swarm and Evolutionary Computation*, p. 101236, 2023.

[39] K. Cho, B. Van Merriënboer, D. Bahdanau, and Y. Bengio, "On the properties of neural machine translation: Encoder-decoder approaches," *arXiv preprint arXiv:1409.1259*, 2014.

[40] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.

[41] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[42] 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," *Journal of Intelligent Manufacturing*, vol. 31, no. 4, pp. 1051–1078, 2020.

[43] X. Jiang, Y. Song, and L. Xing, "Dual-population artificial bee colony algorithm for joint observation satellite mission planning problem," *IEEE Access*, vol. 10, pp. 28 911–28 921, 2022.

[44] Z. Zhou, E. Chen, F. Wu, Z. Chang, and L. Xing, "Multi-satellite scheduling problem with marginal decreasing imaging duration: An improved adaptive ant colony algorithm," *Computers & Industrial Engineering*, vol. 176, p. 108890, 2023.
[45] M. Hollander, D. A. Wolfe, and E. Chicken, *Nonparametric statistical methods*. John Wiley & Sons, 2013.

Yanjie Song received double B.S. degrees in Management from Tianjin University, Tianjin, China, in 2017 and the Ph.D. degree in Engineering from National University of Defense Technology, Changsha, China, in 2023. He has published more than 40 papers in Swarm and Evolutionary Computation, Information Sciences, Computers & Industrial Engineering and other journals. He has authored 1 academic book, obtained 5 National Invention Patents, and hosted/participated in more than 15 projects.

His research interests include computational intelligence, evolutionary algorithm, combinatorial optimization, and deep reinforcement learning.

He is now the reviewer of the IEEE Trans. on Aerospace and Electronic Systems, Swarm and Evolutionary Computation, Neural Networks, Knowledge-Based Systems, Information Science, Applied Soft Computing, Neural Computing and Applications and more than 20 other journals.

Junwei Ou received the B.Sc. degree from the School of Computer and Science Information Engineering, Henan University, in 2015, and the M.Sc. degree in computer science from the College of Information Engineering, Xiangtan University, in 2019. He is currently working toward the Ph.D. degree in control science and engineering with the College of Systems Engineering, National University of Defense Technology.

His research interests include evolutionary computation, multiobjective optimization, and resource scheduling.

Ponnuthurai Nagaratnam Suganthan (Fellow, IEEE) received the B.A. and M.A. degrees from the University of Cambridge, Cambridge, U.K., and the Ph.D. degree in computer science from Nanyang Technological University, Singapore. Since August 2022, he has been with KINDI Centre for Computing Research, Qatar University, as a research professor.

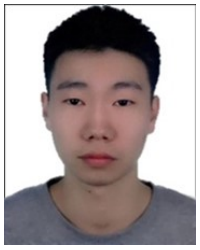
His research interests include randomization-based learning methods, swarm and evolutionary algorithms, pattern recognition, deep learning and applications of swarm, evolutionary & machine learning algorithms.

Dr. Suganthan was the recipient of the IEEE Transactions on Evolutionary Computation Outstanding Paper Award in 2012 and the Highly Cited Researcher Award by the Thomson Reuters in computer science in 2015 . He is currently an Associate Editor for the IEEE Transactions on Evolutionary Computation, the IEEE Transactions on Cybernetics, Information Sciences, and Pattern Recognition and the Founding Co-Editor-in-Chief for ***Swarm and Evolutionary Computation*** Journal.

Witold Pedrycz (IEEE Life Fellow) is Professor in the Department of Electrical and Computer Engineering, University of Alberta, Edmonton, Canada. He is also with the Systems Research Institute of the Polish Academy of Sciences, Warsaw, Poland. Dr. Pedrycz is a foreign member of the Polish Academy of Sciences and a Fellow of the Royal Society of Canada. He is a recipient of several awards including Norbert Wiener award from the IEEE Systems, Man, and Cybernetics Society, IEEE Canada Computer Engineering Medal, a Cajastur Prize for Soft Computing from the European Centre for Soft Computing, a Killam Prize, a Fuzzy Pioneer Award from the IEEE Computational Intelligence Society, and 2019 Meritorious Service Award from the IEEE Systems Man and Cybernetics Society.

His main research directions involve Computational Intelligence, Granular Computing, and Machine Learning, among others.

Professor Pedrycz serves as an Editor-in-Chief of ***Information Sciences***, Editor-in-Chief of ***WIREs Data Mining and Knowledge Discovery*** (Wiley), and Co-editor-in-Chief of ***Int. J. of Granular Computing*** (Springer) and ***J. of Data Information and Management*** (Springer).



Qinwen Yang received the B.S. degree in Engineering from Taiyuan University of Technology, Taiyuan, China, in 2017 and an M.S. degree in Engineering from North Minzu University, Yinchuan, China, in 2023. He is pursuing a doctorate at the University of Auckland, Auckland, New Zealand.

His research interests include satellite mission planning, evolutionary algorithms, deep reinforcement learning, and lifelong learning.

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 Electronic Engineering, Xidian University, China.

His research interests include intelligent optimization methods and scheduling of resources.