IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 21 May 2024, accepted 4 July 2024, date of publication 8 July 2024, date of current version 18 July 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3425161

RESEARCH ARTICLE

Rapid Sequence Generation for Active Debris Removal Mission Based on Attention Mechanism and Pointer Network

ZHIJUN CHEN[1], YUZHU BAI[1], YONG ZHAO[1], AND XIAOQIAN CHEN[2]
[1]College of Aerospace Science and Engineering, National University of Defense Technology, Changsha 410073, China
[2]Chinese Academy of Military Science, Beijing 100091, China

Corresponding author: Yuzhu Bai (baiyuzhu06@nudt.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 12002383, and in part by the Postgraduate Scientific Research Innovation Project of Hunan Province with under Project QL20220006.

ABSTRACT This paper proposes a method based on the attention mechanism combined with a pointer network for rapidly generating sequences of Active debris removal (ADR) missions. Initially, the paper introduces the problem and modeling of ADR missions, establishing the multi-target planning problem as a time-dependent traveling salesman problem (TDTSP) considering the time dimension. Subsequently, during the spacecraft transfer process, it considers using continuous-thrust transfer, utilizing Pontryagin's minimum principle to transform the fuel-optimal continuous-thrust transfer problem into a two-point boundary value problem (TPBVP), then rapidly generating a continuous-thrust dataset to train a surrogate model for evaluating continuous-thrust cost. Furthermore, PN-attention is proposed, which is derived from a pointer network (PN) model that incorporates attention computation, aimed at learning the sequence planning of ADR problems. In the validation and discussion section, PN-attention's performance in solving ADR problems is compared with various heuristic algorithms, verifying its effectiveness, optimality, generalizability, and solution speed.

INDEX TERMS Pointer network, attention mechanism, active debris removal, continuous thrust, time-dependent traveling salesman problem.

I. INTRODUCTION

With the proliferation of space activities, the increasing number of debris in Low Earth Orbit (LEO) could lead to the occurrence of the Kessler Syndrome [1], a chain collision effect that would make the space environment extremely hazardous and could even render certain orbital regions unusable. Therefore, developing and implementing effective LEO debris removal technologies has become an urgent priority. It will not only protect existing space assets but also maintain the sustainability of the space environment, and ensure the long-term development of human space activities. The emergence of Active Debris Removal (ADR) technologies [2], [3] can extend the lifespan of spacecrafts, reduce the need for evasive maneuvers, thereby saving costs and enhancing the safety and efficiency of space operations.

With a large number of potential space debris serving as potential targets that need to be removed, if other factors do not limit the choice of targets, it is often possible to select those that require minimal changes to orbital elements to reduce the cost of transfer in terms of propellant and duration. Since space debris is in orbital motion, ADR missions are generally formulated as a time-dependent traveling salesman problem (TDTSP) [4]. Depending on the number of service spacecrafts involved, these can be single-servicer missions [5] [6], or multiple servicer missions [7]. Based on the transition of service spacecraft, missions can be categorized as multi-impulse ADR missions [8] and continuous-thrust ADR missions [9], [10].

Electric propulsion was more suitable for missions to multiple targets because of its large specific impulse. There are

The associate editor coordinating the review of this manuscript and approving it for publication was Rosario Pecora.

generally direct methods [11], [12] and indirect methods [13], [14] for solving electric-propulsion continuous-thrust transfer problems. Direct methods typically use pseudospectral methods to replace nonlinear problems with high-order interpolation, which are computationally efficient but limited in accuracy. Indirect methods [15], based on Pontryagin's Minimum Principle, transform the continuous-thrust problem into a two-point boundary value problem (TPBVP). The advantage of indirect methods is that the solutions obtained have higher accuracy, satisfying first-order optimality conditions. However, the solution depends on the initial guess of the costate, which lacks a clear physical meaning, and is complex and sensitive to boundary conditions and constraints. To address this issue, the costate can be treated as parameters to be optimized, using nonlinear programming or intelligent algorithms [16]. Jiang et al. [17] used homotopy methods to transform the fuel-optimal continuous-thrust problem into an energy-optimal form, avoiding the singularities caused by control discontinuities, and obtained high-precision solutions using Particle Swarm Optimization (PSO) and Newton's iteration. In missions similar to ADR, when considering the use of continuous thrust as the propulsion for the service spacecraft, it is necessary to solve high-precision, continuous-thrust problems using indirect methods.

However, the Newton's method for solving nonlinear equations is slow and unsuitable for the rapid calculation of continuous-thrust trajectories, making it difficult to quickly provide fuel consumption costs for ADR, especially when facing a large scale of targets. A method for quickly computing near-optimal transfers between low-eccentricity orbits is introduced [9], offering accurate cost and duration estimates, demonstrating that it enables the rapid design of favorable missions. Zuiani and Vasile [18] proposed a predefined pattern to minimize continuous-thrust propellant consumption and transfer time for multi-objective optimization problems involving the sequence and timing of the removal of five hypothetical pieces of debris. Li et al. [19] proposed a rapid and effective method for solving optimal J2-perturbed multi-debris removal problems, with 75 pieces of debris being removed within 327 days. Jorgensen and Sharf [20] solved the minimum-time continuous-thrust orbital transfer problem implemented in GPOPS-II, selecting a set of five pieces of debris with small inclination differences. Narayanaswamy et al. [21] developed a two-step procedure combining relevant distance metrics to approximate the transfer cost and RQ-Law to generate continuous-thrust rendezvous trajectories in the determined order. Shen et al. [22] determined the flight sequence in a ballistic scenario, using a differential evolution method with constructing a three-impulse transfer problem, then implementing local optimization with continuous-thrust propulsion based on the solutions of impulsive trajectories. Zona et al. [23] presented an evolutionary optimization algorithm for ADR, optimizing the sequence of debris capture and testing various operators to enhance search capabilities. The aforementioned research can provide reasonable approximations for continuous-thrust costs, but it employs a mission planning approach based on heuristic local search algorithms. Each time a new problem is encountered, such methods typically require restarting the search process or adjusting heuristic rules to achieve better results, which is computationally expensive. Moreover, when dealing with large-scale problems, extensive iterative searches can still lead to significant computational time, and the approximate methods remain challenging to extend to online, real-time optimization problems.

Machine learning, especially Deep Neural Networks (DNN), has advantages in approximating quantifying uncertainties, and establishing high-precision nonlinear dynamic models [24]. Considering the requirements for robustness and real-time performance in space exploration missions, DNN has been proven to be a practical method. For the optimal transfer problem, many scholars have conducted related research using intelligent methods in recent years. Chen et al. [25] used DNN to learn the mapping from initial states to costate solutions, enabling rapid solving of continuous-thrust problems and quickly providing transfer costs for mission planning problems. Li et al. [8] proposed feed-forward DNN to achieve rapid and accurate estimation of time-optimal and fuel-optimal continuous-thrust and multi-pulse transfers. Viavattene et al. [26] used DNN to quickly estimate the cost and duration of de-orbiting transfers for a range of debris objects, identifying the optimal sequence of objects that minimize mission costs. Guo et al. [27] used DNN to estimate the minimum transfer time between every two asteroids and modified beam search to optimize the sequences of multi-asteroid missions. Xu et al. [28] proposed a DNN-based estimation method for approximating the optimal velocity increments of perturbed multiple-impulse rendezvous and a reinforcement learning (RL) method for optimizing the sequence of debris removal and rendezvous timing. Yang et al. [29] combined RL formulation with a modified upper confidence bound tree search algorithm for the ADR planning task.

As an important branch of deep learning, Deep Reinforcement Learning (DRL) [30] is mainly used to solve sequential decision-making problems, making action choices based on the current environmental state, and continuously adjusting its own strategy according to the feedback of actions, so as to achieve the set goals. Combinatorial optimization problems involve optimal selection of decision variables within a discrete decision space, which naturally shares similar characteristics with the 'action selection' in RL, and the 'offline training, online decision-making' feature of DRL makes it possible to solve combinatorial optimization problems in real-time online. Multi-target ADR sequence planning can be regarded as a TPTSP, which itself is a combinatorial optimization problem. Currently, combinatorial optimization methods based on DRL are mainly divided into two categories: end-to-end algorithms based on DRL and local search

improvement algorithms based on DRL. The aforementioned local search improvement algorithms based on DRL need to start searching from scratch when encountering a new task scenario. The end-to-end method based on DRL does not require searching and can directly outputs the solution to the problem, offering the advantage of fast solution speed.

The Pointer Network (PN) is a typical end-to-end DRL algorithm. Vinyals et al. [31] first proposed the PN model for solving TSP problems and used a large number of 'TSP city coordinates-optimal path' samples to offline train the parameters of the PN by supervised learning. The trained PN model can solve TSP problem with the same distribution characteristics as the training set and achieves rapid solution of small-scale TSP problems with up to 50 cities, significantly improving computational efficiency. PN uses an encoder based on a DNN to encode the input sequence of the combinatorial optimization problem (such as city coordinates in TSP), and then calculates the selection probability of each node through a decoder and an attention mechanism [32]. The nodes are selected step by step in an autoregressive manner until a complete solution (such as the order of city visits) is obtained.

However, above training in a supervised manner leads to the quality of the solution not exceeding the quality of the sample solution, and constructing training samples in advance takes a lot of time, limiting its application to solving larger-scale combinatorial optimization problem. In view of this, Bello et al. [33] used RL to train the PN, treating each problem instance as a training sample, using the objective function of the problem as feedback, and training with REINFORCE algorithm. This model surpassed the supervised training model obtained by Vinyals et al. [31] and could approach the optimal solution on the 100-city TSP problem, showing good generalization. Furthermore, Kool et al. [34] achieved the better optimization effect on TSP problems below 100 scale.

In summary, ADR problem that considers continuous thrust includes two-level optimization, that is the sequential planning of multiple targets and the point-to-point continuous-thrust transfer trajectory optimization. The overall planning of the ADR problem involves both continuous and integer variables, resulting in a vast search space. Traditional heuristic algorithms are slow when solving ADR problems, and they are not suitable for on-orbit ADR planning. The aforementioned PN-based method is equally applicable to ADR problems. Utilizing the PN approach, once the model is trained, it possesses strong generalization capabilities, allowing for rapid solving of ADR problem instances with similar distribution characteristics. This enables fast, real-time in-orbit computation, offering significant practical value.

To the authors' knowledge, this paper is the first time to introduce a PN method to generate the sequence of in-orbit multi-target ADR using continuous thrust as the propulsion for maneuvers. The main contributions and innovations of this paper are as follows:

(1) A continuous-thrust cost estimator is proposed based on the DNN, which uses datasets generated by the indirect method for network training, and can quickly provide fuel consumption for point-to-point continuous-thrust transfers;
(2) PN-attention sequence planner is introduced for generating the sequences of multi-target ADR. This approach can tackle time-dependent sequence planning challenges and represents the principal innovation of this paper.

The main structure of this paper is as follows: The first part is the introduction, which introduces the main background and methods involved in this paper. The second section elaborates on the ADR mission planning problem. The third section introduces the optimal control of continuous thrust and proposed continuous-thrust cost estimator. The fourth section proposes the pointer network architecture and methods of attention computation. The fifth part presents experiments and compares PN-attention with heuristic algorithms, discussing the advantages in terms of effectiveness, optimality, and generalization performance.

II. PROBLEM FORMULATION OF ADR

This paper focuses on the ADR problem in low Earth orbit, shown in Figure 1. In this paper, it is assumed that the ADR problem proposed involves a single service spacecraft (servicer) rendezvousing with multiple pieces of space debris target to perform in-orbit repairs and install thruster deorbit kits (TDKs) [35] on decommissioned targets. The TDK will then drive this target into a graveyard orbit. Servicer use continuous thrust during transfer between targets.

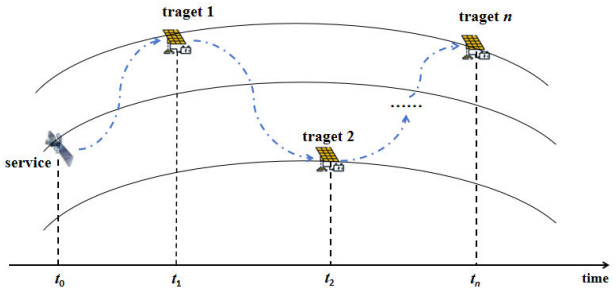

FIGURE 1. Schematic diagram of the ADR mission.

The mission planning problem designed in this paper requires that all rendezvous with targets be completed within the mission time, and the fuel consumption of the servicer would be optimized during the rendezvous process. The model of the ADR problem addressed can be defined as follows:

$$\begin{aligned} &\text{find} \quad \boldsymbol{t}, \boldsymbol{s} \\ &\min \quad J = \sum |\Delta m_i| \end{aligned} \tag{1}$$

where, $J = \sum |\Delta m_i|$ is the optimization objectives, and Δm_i represents the servicer's consumed mass of fuel during the rendezvous transfer with the i-th target. $\boldsymbol{t} = (t_1, t_2, \ldots, t_n)$

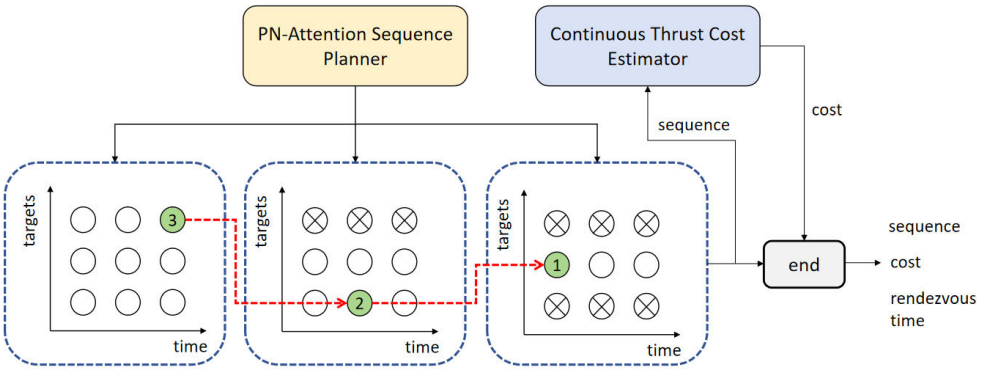

FIGURE 2. Flowchart of the ADR mission based on PN-attention sequence planner and continuous-thrust cost estimator.

is the time sequence, $\boldsymbol{t}_i = (t_i, \Delta t_i)$, and t_i is time instance of rendezvous with servicer and the i-th space target, Δt_i is the transfer time before the rendezvous with the i-th target. $\boldsymbol{s}$ is the planning sequence.

The aforementioned model is a type of mixed-integer programming problem, which includes both discrete variables, such as sequences, and continuous variables, such as departure times and transfer times. Traditional methods for solving the ADR sequence planning problem include heuristic algorithms and other local search techniques. These methods plan for a single mission scenario of ADR, but will restart the planning process when facing new scenarios.

This paper, in combination with current artificial intelligence algorithms, proposes a sequence planner based on the PN model, hoping to achieve rapid generation for on-orbit service sequence. It is assumed that the propulsion of the servicer is of a continuous-thrust mode. In the ADR planning problem, it is generally divided into two-level planning models: the upper-level model plans the sequence, and the lower-level model optimizes continuous-thrust transfer trajectory, where lower-level will return the fuel consumption cost to the upper-level. Generally, the lower-level needs to seek optimization in each iteration. The upper-level sequence planning is an NP-hard problem and will lead to a large computational workload when the lower-level trajectory optimization needs to be invoked at each step in the upper-level sequence planning process. Especially for continuous-thrust problem, the solution of fuel-optimal trajectory generally utilizes the indirect method, transforming the TPBVP into a nonlinear programming problem. This method has a large computational workload, small convergence domain, and it is generally difficult to obtain a feasible solution.

The overall mission planning process flowchart of this paper is shown in Figure 2. Based on the author's previous work [25], we will use DNN to train a continuous-thrust cost estimator to map the inputs and outputs for rapidly obtaining the optimal fuel consumption. This result is then passed to the upper-level sequence planner, which can significantly enhance computational efficiency. The upper-level mission planner also incorporates current artificial intelligence semantic models, PN method combined with attention mechanisms, which can quickly solve sequence-to-sequence (Seq2Seq) mapping problems and avoid the issue of gradient vanishing that occurs in recurrent neural networks (RNN), thereby training a sequence planner.

III. INTRODUCTION TO CONTINUOUS THRUST COST ESTIMATOR

Orbital maneuvers under continuous thrust can rapidly accomplish space missions in complex processes, but also present numerous challenges. The continuous-thrust trajectory optimization problem is characterized by strong nonlinearity and multiple peaks, making optimization quite difficult. The continuous-thrust fuel-optimal control problem aims to minimize fuel consumption during the control process. To solve this problem, traditional methods primarily employ indirect methods from optimal control theory, establishing a system of nonlinear equations using Pontryagin's minimum principle and solving for the optimal solution with shooting methods or collocation methods.

A. DESCRIPTION OF CONTINUOUS-THRUST FUEL-OPTIMAL CONTROL

Initially, a modified equinoctial orbital element is adopted for description, establishing the dynamic model for spacecraft orbital transfer. When the classical orbital elements $\boldsymbol{O} = [a, e, i, \Omega, \omega, M_0]^{\mathrm{T}}$ of a spacecraft are known, they can be converted to the modified equinoctial orbital elements $\boldsymbol{x} = [p, f, g, h, k, L]^{\mathrm{T}}$ as follow:

$$\begin{cases} p = a\left(1 - e^2\right) \\ f = e\cos(\Omega + \omega) \\ g = e\sin(\Omega + \omega) \\ h = \tan(i/2)\cos\Omega \\ k = \tan(i/2)\sin\Omega \\ L = \Omega + \omega + M_0 \end{cases} \tag{2}$$

where, a is semimajor axis of orbit, e is eccentricity, i is inclination, Ω is right ascension of the ascending node

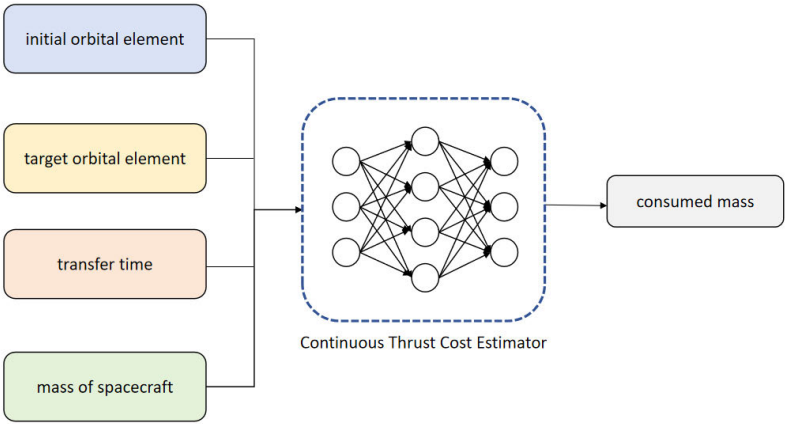

FIGURE 3. Architecture of the continuous-thrust cost estimator network.

(RAAN), ω is argument of perigee, and M_0 is initial mean anomaly. $[p, f, g, h, k, L]$ corresponds to the parameters of modified equinoctial orbital elements. The parameters mentioned above, both $\boldsymbol{O}$ and $\boldsymbol{x}$, are used to describe the absolute motion of a spacecraft in Earth's orbit.

Referring to [36], we can derive the differential equations for the modified equinoctial orbital elements, considering the impact of J2 perturbation in the differential equations. Apart from fuel and time constraints, some considerations have been made regarding the achievable changes in RAAN, given the required altitude and inclination alterations for the transfer:

$$\dot{\Omega} = -\left[\frac{3}{2}\frac{J_2\sqrt{\mu}R_e^2}{a^{7/2}\left(1-e^2\right)^2}\cos(i)\right] \tag{3}$$

where, μ is the Earth's gravitational constant, R_e is the Earth's equatorial radius, and J_2 is the second-order harmonic coefficient of the Earth's oblateness.

The objective J_{opt} of fuel-optimal control for continuous thrust can be described as follows:

$$J_{opt} = \lambda_0 \frac{T_{\max}}{I_{\mathrm{sp}} g_0} \int_{t_0}^{t_f} u dt \tag{4}$$

where λ_0 is the costate variable introduced by the minimum principle, $T_{\max}$ is the maximum thrust, I_{sp} is the specific impulse, and g_0 is the gravity acceleration at sea level. u represents the continuous-thrust control input of the spacecraft, which is a continuous quantity. By integrating u over the mission time, we can obtain the fuel consumption of the spacecraft during the continuous-thrust transfer.

If the orbital dynamics of servicer can be described by the following equation:

$$\dot{\boldsymbol{x}} = \frac{T_{\max} u}{m}\boldsymbol{M}(\boldsymbol{x}) + \boldsymbol{D}(\boldsymbol{x}) \tag{5}$$

then based on Pontryagin's minimum principle, the above optimal control problem can be transformed into a problem of solving the extremum of the Hamiltonian function. The Hamiltonian function for the fuel-optimal control problem can be proposed as follows [17]:

$$H = \lambda_x^T \boldsymbol{M}(x)\frac{T_{\max} u}{m} + \lambda_{\boldsymbol{x}}^T \boldsymbol{D}(\boldsymbol{x}) - \lambda_m \frac{T_{\max} u}{I_{\mathrm{sp}} g_0} + \lambda_0 \frac{T_{\max} u}{I_{\mathrm{sp}} g_0} \tag{6}$$

where, λ_x and λ_m are the adjoint constates associated with the state and mass, respectively. $\boldsymbol{M}(\boldsymbol{x})$ and $\boldsymbol{D}(\boldsymbol{x})$ are the state matrices corresponding to the dynamic differential equations.

For the costate differential equations, they can be obtained by taking partial derivatives $\dot{\lambda}_x = -H/\boldsymbol{x}$ of the Hamiltonian function (6), detailed in the appendix of paper [36].

Referring to [17], it is known that in order to find the extremum of the Hamiltonian function in equation(6), the control variable u is influenced by the following switch function:

$$u = \begin{cases} 0, & \text{if } \rho > 0 \\ 1, & \text{if } \rho < 0 \\ [0,1], & \text{if } \rho = 0 \end{cases}$$
$$\rho = 1 - \frac{I_{\mathrm{sp}} g_0 \left\| \boldsymbol{M}^T \lambda_x \right\|}{\lambda_0 m} - \frac{\lambda_m}{\lambda_0} \tag{7}$$

As seen from equation(7), the control variable u switches between 0 and 1, it is not a continuously varying quantity and has points of discontinuity, making it difficult to obtain gradients. This further complicates the use of Newton or Powell method for solving nonlinear equations. When using Newton method, the Jacobian matrix of the equations needs to be calculated, and any discontinuities in the equations can lead to singularities during the solution process. According to [17], homotopy methods are introduced, mapping the fuel-optimal problem to an energy-optimal problem, thus mitigating the effects of discontinuities.

B. CONTINUOUS-THRUST COST ESTIMATOR

This section will introduce the neural network architecture of the continuous-thrust cost estimator and the design method of the loss function.

The neural network shown in Figure 3 is a fully connected neural network, with a 14-dimensional input consisting of

the orbital elements of the departing spacecraft $\boldsymbol{O}_{\mathrm{d}}$ and the arriving spacecraft $\boldsymbol{O}_{\mathrm{a}}$, as well as the mass m_0 of the servicer at departure time and the orbit transfer time Δt. The network outputs a 1-dimensional fuel consumption cost Δm_i. To address the problem of rapid solution finding for continuous-thrust fuel-optimal control, the generated dataset would use the concept of the neighboring point iteration algorithm (NPIA) proposed in earlier work [25], enabling a rapid solution process. Then the dataset of fuel-optimal control will be used to train the DNN-based continuous-thrust const estimator.

ReLU is used as the activation function between each layer. In order to effectively fit the dataset, it is crucial to devise a good loss function, which will greatly influence the regression parameters. This paper employs the Mean Absolute Error (MAE) as the loss function of DNN. The loss function of MAE is:

$$L_{\mathrm{MAE}} = \frac{1}{N}\sum_{i=1}^{N}\left|\hat{y}_i - y_i\right| \tag{8}$$

where, $\hat{y}_i$ represents the predicted output of the neural network, and y_i is the true value of the generated sample.

Using the well-trained DNN, we can input a set of samples within the distribution range of the departure and arrival orbital elements, as well as the initial mass of the servicer and the transfer time. The DNN can then quickly output the fuel consumption solution for the optimal control problem, thereby rapidly providing the required consumed cost indicators for ADR planning problems.

IV. POINTER NETWORK FOR ADR SEQUENCE PLANNER

This section will explore Seq2Seq-based PN models, a method designed to provide modeling capabilities for combinatorial optimization problems with sequential attributes. First, PN processes the input sequence in the combinatorial optimization problem through an encoder, transforming it into a feature vector that encapsulates the input information. Then, a decoder decodes this feature vector and combines it with an innovative attention mechanism to construct solutions element by element in an autoregressive manner, that is, adding an element to the current partial solution at each step until a complete solution is formed. This section will utilize this method to solve the sequence planning for ADR.

A. INTRODUCTION TO POINTER NETWORKS AND ATTENTION MECHANISM

Specifically, define a combinatorial optimization problem $P = P_1, P_2, \cdots, P_n$ with solution $C^P = C_1, \cdots, C_{m(P)}$. This problem can be modeled as a mapping from the sequence of problems P to the sequence of solutions C^P, thus requiring the construction of a DNN model with parameters θ to estimate the probability distribution $p\left(C^P \mid P; \theta\right)$ of this mapping. Specifically, the chain rule for conditional probabilities can be used to decompose:

$$p\left(C^P \mid P; \theta\right) = \prod_{m(P)}^{i=1} p_\theta\left(C_i \mid C_1, \ldots, C_{i-1}, P; \theta\right) \tag{9}$$

The parameters of the neural network model are optimized by maximizing the conditional probability:

$$\theta^* = \arg\max_{\theta} \sum_{P, C^P} \log p\left(C^P | P; \theta\right) \tag{10}$$

Regarding the modeling process described above, a PN model can be used to achieve the Seq2Seq mapping. Similar to Seq2Seq models, PN consists of an encoder, decoder, and attention mechanism, with both the encoder and decoder being RNNs, which are used to model $p_\theta\left(C_i \mid C_1, \ldots, C_{i-1}, P; \theta\right)$.

The encoder RNN is responsible for processing the node coordinates, generating a corresponding hidden state $e_1, \cdots, e_n$ for each node, and outputting a final fixed-length vector. The decoder RNN then takes this fixed-length vector and generates the current hidden layer state $d_1, \cdots, d_{m(P)}$ at each decoding step. At each decoding step, the model calculates the attention weights between the current decoder hidden layer state d_i and all encoder hidden states $e_1, \cdots, e_n$, resulting in a weighted hidden layer state that guides the output of the sequence. The attention calculation formula is as follows:

$$\begin{aligned} u_j^i &= v^T \tanh\left(W_1 e_j + W_2 d_i\right) \\ a_j^i &= \mathrm{softmax}\left(u_j^i\right), \quad j \in (1, \ldots, n) \\ d_i' &= \sum_{j=1}^{n} a_j^i e_j \end{aligned} \tag{11}$$

where, the softmax function normalizes the output attention values, v, W_1, W_2 represents the neural network parameters to be optimized, and the weighted encoding hidden states e_j are concatenated with the decoder hidden states d_i.

However, when dealing with problems like ADR, traditional Seq2Seq mapping methods encounter obstacles. This is because the solution length of ADR problems is related to the length of the input sequence, and the output sequence is usually a permutation of the node numbers in the input sequence. Therefore, standard Seq2Seq models, as described in equation (11), are not directly applicable to these types of problems.

To overcome this limitation, the PN model adjusts and optimizes the traditional method of attention calculation. Through this improvement, the PN can effectively address and solve combinatorial optimization problems. Specifically, instead of simply generating a fixed output sequence, the PN dynamically selects elements from the input sequence to construct solutions, thus adaptively generating output sequences of varying lengths based on the characteristics of the input

data. The PN model uses equation(12) to calculate the conditional probability:

$$u_j^i = v^T \tanh\left(W_1 e_j + W_2 d_i\right) j \in (1, \ldots, n)$$
$$p\left(C_i \mid C_1, \ldots, C_{i-1}, P\right) = \text{softmax}\left(u^i\right) \tag{12}$$

The novel PN employs a modified version of traditional attention mechanism. In this variant, the model forgoes using the third equation in (11) to further process the attention-weighted encoder hidden states. Instead, the PN directly uses the attention weights calculated from the second equation as 'pointers' which indicate specific positions in the input sequence. Through this method, the PN achieves an optimized 'combinatorial' process over the elements of the input sequence and ultimately outputs a permutation of node numbers. This approach effectively addresses combinatorial optimization problems where the solution is variable and dependent on the content of the input sequence.

The parameters of PN are determined by maximizing the conditional probability $p\left(C^P \mid P; \theta\right)$:

$$\theta^* = \arg\max_{\theta} \sum_{P, C^P} \log p\left(C^P \mid P; \theta\right) \tag{13}$$

Regarding the ADR problem, which is essentially a multi-target rendezvous problem with distinct sequential characteristics, and where the state of nodes dynamically changes according to orbital motion over time in the decision-making process, the above-mentioned solution framework is suitable for modeling this problem using a PN architecture. Moreover, it is necessary to incorporate mechanisms for handling dynamic characteristics into the design of the neural network model structure.

For the ADR problem s with n debris targets, the parametric representation of the model is as follows: In the ADR problem, the characteristics $\mathbf{x}_i$ of the debris targets i are defined by their orbital positions, and the solution $\pi = (\pi_1, \ldots, \pi_k, k \leq n)$ is a combination of serial numbers of subsets of debris. The solution must satisfy that all targets should be visited. To solve the ADR problem, it is necessary to design a NN model with parameters θ, which takes the orbital elements of targets and rendezvous times of the ADR problem s as input, and outputs the sequence π and rendezvous times t_i. This model strategy is defined as $p(\pi \mid s)$, which can achieve the mapping from s to π:

$$p_\theta(\pi \mid s) = \prod_{k}^{t=1} p_\theta\left(\pi_t \mid s, \pi_{1\sim t-1}\right), k \leq n. \tag{14}$$

Since the probability of selecting the next target is closely related to the information of the previously selected debris targets $\pi_{1\sim t-1}$, it has distinct sequential characteristics—the PN model architecture is used to model the ADR problem. Specifically, an attention model is introduced, and a static embedding based on a multi-head attention mechanism is used in the encoder to handle the static features of the problem. In the decoder, the design is reconceptualized by combining a RNN with an attention mechanism to enhance model performance. Next, a detailed introduction to the structure of this PN model will be provided.

B. PN-ATTENTION MODEL FOR ADR

To model the ADR problem using the PN approach, it is first necessary to process the problem's input to fit the input structure of a DNN. Given the input x_i for target nodes i, the input information includes the orbital elements information of the targets to be visited and the discretized rendezvous times. Assuming the range of rendezvous time is $t_{\min} \leq \Delta t_i \leq t_{\max}$, the rendezvous times can be discretized into N time points within this time range, with the time interval between each segment being T_r.

Furthermore, map $\boldsymbol{x}_i$ to a high-dimensional vector $\mathbf{X}_i$, specifically defining the initial feature vector of node i as a high-dimensional vector of dimension d_h, which is the linear mapping of node i's problem input to d_h dimensions:

$$\mathbf{X}_i = \mathbf{W}^{\mathbf{x}} \boldsymbol{x}_i \tag{15}$$

where, $\mathbf{W}^{\mathbf{x}}$ represents the weight parameters, and by setting $d_h = 128$. This process achieves the initial extraction of node feature vectors through a linear mapping based on a fully connected neural network.

The PN-attention model adopted in this paper is shown in Figure 4. Its basic structure consists of an encoder and a decoder connected in series. The encoder is used to embed nodes into vectors, while the decoder is used to decode vectors into output sequences. Both the encoder and decoder are neural networks, with the decoder being RNN. When using the encoder to encode the input i, a hidden layer vector $\boldsymbol{e}_i$ is generated for orbital position input of each target. Here, $\boldsymbol{e}_i$ can be interpreted as the feature vector of target i. The output of the encoder is a vector. As shown in Figure 4, the vector is used as the input to the decoder of RNN, and the decoding process proceeds sequentially. The vector serves as the initial hidden layer vector $\boldsymbol{d}_0$, which is used to select the first target to visit. The next hidden layer vector $\boldsymbol{d}_1$ is computed by the RNN of the decoder. In decoding step t, the probability of selecting a target is calculated based on the encoder state $\boldsymbol{d}_{t-1}$ and the feature vector e_i of each target. The RNN reads the node feature vector of the selected target and outputs the current decoder state $\boldsymbol{d}_t$, which is then used in the next decoding step. The calculation of the target decoding probability is computed as follows.

$$u_i^t = \boldsymbol{v}^T \tanh\left(\boldsymbol{W}_1 \boldsymbol{e}_i + \boldsymbol{W}_2 \boldsymbol{d}_t\right) i \in (1, \ldots, n)$$
$$P\left(y_{t+1} \mid y_1, \ldots, y_t, \mathbf{X}\right) = \text{softmax}\left(u^t\right) \tag{16}$$

To select the first target to visit in decoding step $t = 1$, it is calculated u_1^1, u_2^1, u_3^1 based on equation(16) using $(\boldsymbol{d}_0, \boldsymbol{e}_1)$, $(\boldsymbol{d}_0, \boldsymbol{e}_2)$, and $(\boldsymbol{d}_0, \boldsymbol{e}_3)$. The one with the highest u^1 is chosen as the first target, that is u_2^1, target **2**. By inputting the node embedding of target **2** into the decoder RNN, $\boldsymbol{d}_1$ can be obtained. In the process of selecting the next spatial target, the selection probability of targets can be obtained by applying

the attention calculation formula on $(\boldsymbol{d}_1, \boldsymbol{e}_1)$, $(\boldsymbol{d}_1, \boldsymbol{e}_2)$, and $(\boldsymbol{d}_1, \boldsymbol{e}_3)$. At this time, the target with the highest probability, target **3**, can be selected. This process continues in a loop until a complete solution is formed. The problem-solving can be realized by training the parameters $\boldsymbol{v}$, $\boldsymbol{W}_1$, $\boldsymbol{W}_2$ offline.

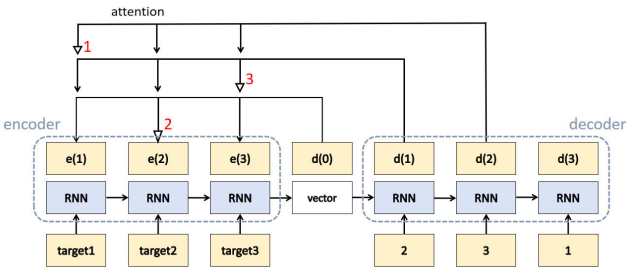

FIGURE 4. The encoder-decoder PN-attention architecture for solving the ADR problem.

In the mentioned PN-attention structure, the primary function of the encoder is to create a feature vector $\boldsymbol{e}_i$ for each target element i. With the evolution of attention mechanisms, the self-attention module has demonstrated its superiority in sequence feature extraction. The self-attention mechanism is a key component of Transformer model [32], which has been widely applied in learning tasks that process sequential data.

To effectively extract the features of the ADR problem, this study incorporates a self-attention mechanism in the design of the encoder. This mechanism generates node feature vectors by calculating the self-attention weights between nodes. Thus, the feature vector of each node contains not only the information of the node itself but also the interactive relationships with other nodes, which is crucial when dealing with ADR problems as it helps to capture the structural patterns of the problem more effectively. Compared to the standard attention computation method(16), the self-attention mechanism employs Scaled Dot-Product Attention, which calculates attention weights through the matrix multiplication $\mathbf{q}_t^T \mathbf{k}_i$ of query values $\mathbf{q}_t$ and key values $\mathbf{k}_i$. The working principle is to evaluate the importance of each node within the input sequence relative to other nodes, and these calculated attention weights serve as the feature vectors for the nodes. As a result, the processed sequence reflects not only the unique features of the nodes themselves but also the complexity of the interrelationships between nodes.

The self-attention mechanism is implemented through the calculation involving query and key-value pairs. Here, the query $\mathbf{q}_i$, key $\mathbf{k}_i$, and value $\mathbf{v}_i$ are all obtained from the input sequence through linear transformations. Self-attention focuses on the internal relationships within the input sequence itself. During the computation, if the corresponding key of a certain value has a higher matching degree with the query, then this value will be assigned more 'attention', meaning it will be given a greater weight in subsequent processing. Such a mechanism enables the model to adaptively highlight important information in the input sequence according to the task requirements. Firstly, compute the query, key, and value, which are the linear transformations of the input sequence:

$$\mathbf{Q} = \boldsymbol{W}^Q \mathbf{X}, \mathbf{K} = \boldsymbol{W}^K \mathbf{X}, \mathbf{V} = \boldsymbol{W}^V \mathbf{X} \tag{17}$$

where $\boldsymbol{W}^Q$, $\boldsymbol{W}^K$, and $\boldsymbol{W}^V$ are the weight values of the fully connected neural network. The query, key, and value for the i_{th} node is:

$$\mathbf{q}_i = \boldsymbol{W}^Q \mathbf{X}_i, \mathbf{k}_i = \boldsymbol{W}^K \mathbf{X}_i, \mathbf{v}_i = \boldsymbol{W}^V \mathbf{X}_i \tag{18}$$

The self-attention value computation process for the i node is shown in Figure 5. Initially, the relational features between the i node and other nodes are calculated by the matching function of the i node's query with the keys of all other nodes. The obtained matching values serve as weights and are normalized using the softmax function. The attention value is then computed as the weighted sum of the values $\mathbf{V}$, where the weights are $\mathbf{q}_i^T \boldsymbol{K}$. Therefore, the attention value for the i node is $\mathbf{q}_i^T \boldsymbol{K} \boldsymbol{V}$, which has the same dimensionality d_h as its initial feature vector. Through the self-attention computation, the attention value of the i node stores not only its own features but also the relational features with other nodes.

The attention values for all nodes are calculated simultaneously through matrix multiplication:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}}\right)\mathbf{V} \tag{19}$$

where, $\sqrt{d_k}$ is the scaling factor, d_k is the dimensionality of key.

During the self-attention computation process, it can be iteratively performed multiple times to produce several feature representations with different characteristics, a process known as Multi-Head Attention. Specifically, the original feature vector of dimension d_h is first split into M sub-vectors of dimension d_h/M. Then, each sub-vector is passed through different linear transformation parameters $\boldsymbol{W}^Q$, $\boldsymbol{W}^K$, and $\boldsymbol{W}^V$ to obtain a specific set of queries, keys, and values. The self-attention mechanism is used to process each group of queries, keys, and values, resulting in M sets of sub-feature vectors. Finally, these sub-feature vectors are concatenated to form a composite node feature vector of dimension d_h, which is the same as the original feature vector. This is the computational method of multi-head attention. By applying the multi-head attention mechanism in the encoder, more diverse features can be extracted from the ADR problem, thereby more accurately capturing the complex structural patterns of the problem.

C. CONDITIONAL PROBABILITIES THROUGH ATTENTION MECHANISM

The above content introduced the construction of queries and keys. Based on the query $\mathbf{q}_t$ at step t and the key values of all nodes, one can compute the probability of debris target selection at step t. Among all $\boldsymbol{k}_i, i = 1 \ldots n$, the one that best matches $\mathbf{q}_t$ will be chosen as the next target to visit. This computation process is realized through the attention mechanism. Specifically, during the decoding process at step t, the

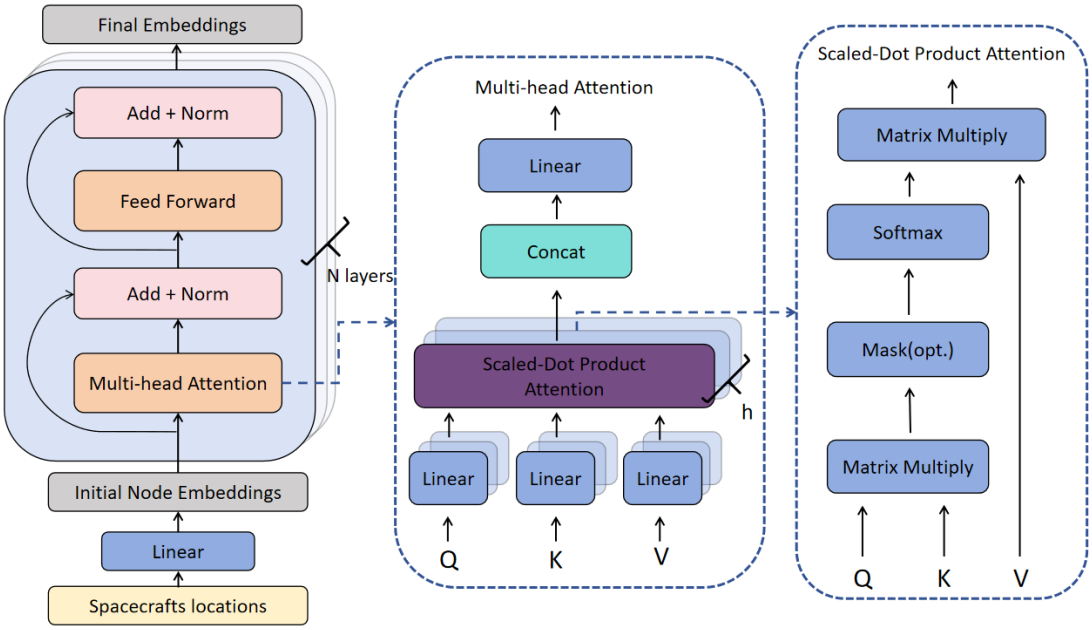

FIGURE 5. Computational processes of the multi-head attention mechanism and self-attention mechanism.

Scaled Dot-Product Attention method is used to calculate the selection probabilities of all targets based on query-key:

$$u_i^t = \frac{\mathbf{q}_t^T \mathbf{k}_i}{\sqrt{d_k}} \tag{20}$$

A mask mechanism is employed to prevent targets that have already been visited from being selected again:

$$u_i^t = \begin{cases} -\infty & \text{if } j \in \pi_{1\sim t-1} \\ \dfrac{\mathbf{q}_t^T \mathbf{k}_i}{\sqrt{d_k}} & \text{otherwise} \end{cases} \tag{21}$$

After regularizing the above probabilities with the softmax operator, the selection probability of the chosen spatial debris becomes zero. The final conditional probability of target selection is realized through the following softmax operation:

$$p_i = \frac{\exp\left(u_i^t\right)}{\sum_{j=1}^{n} \exp\left(u_j^t\right)}, i = 1, \cdots, n \tag{22}$$

When using a PN-attention model for ADR problem, decisions in the decoding stage can be based on two different strategies: the greedy strategy and the sampling strategy. In the greedy strategy, each step of the decoding process t selects the node p_i with the current highest probability for visiting. This strategy aims to make locally optimal decisions at each step in hopes of reaching a global optimum. During the model deployment phase, that is, when applied online, the greedy strategy is usually adopted because it can quickly determine the most probable node at each decision point, thereby speeding up the decision-making process and improving operational efficiency.

In summary, the PN-attention model designed in this section, tailored for the characteristics of the ADR problem, consists of an encoder, a decoder, and an attention mechanism. The encoder is responsible for extracting features of the ADR problem as input and outputting the feature vectors for each node. The decoder then carries out the decoding process in a certain order until all targets have been visited. In this process, the selected nodes ultimately constitute the solution $(\pi_0, \pi_1, \cdots)$ to the problem.

D. REINFORCE TRAINING

Given an instance of a ADR problem s, the previous section defined a PN model parameterized by θ, which can generate a conditional probability distribution $p_\theta(\pi \mid s)$ through equations (21) and(22). A solution $\pi \mid s$ can be obtained by sampling from this conditional probability distribution $p_\theta(\pi \mid s)$.

To minimize total fuel consumption of ADR, it is suitable to use the REINFORCE algorithm [37] based on episodes for training the aforementioned PN-attention model. This algorithm uses an episode-based Monte Carlo method to calculate reward values, and it keeps executing the policy until a complete solution is constructed, at which point the negative of the total fuel consumption is calculated as the reward. Based on this definition, the policy parameters can be continuously updated through state-action-reward.

Given an action $\pi \mid s$ and the corresponding loss function $L(\pi)$, one can update the model parameters θ via gradient descent based on the REINFORCE algorithm:

$$\begin{aligned} \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}) &= \mathbf{E}_{p_{\boldsymbol{\theta}}(\pi|s)} \left[\nabla \log p_{\boldsymbol{\theta}}(\boldsymbol{\pi} \mid s) L(\boldsymbol{\pi}) \right] \\ \boldsymbol{\theta} &\leftarrow \boldsymbol{\theta} + \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}). \end{aligned} \tag{23}$$

The construction of ADR solutions requires sampling a large number of actions from the probability distribution, with total fuel consumption serving as the reward. Due to the uncertainty of sampling and the numerous sampling processes within each episode, the rewards obtained in different episodes can have high variance. This variance can lead to conflicting gradient descent directions for the policy parameters, affecting convergence speed.

To reduce the variance in policy parameter updates, a baseline value $b(s)$ is typically introduced to rewrite the policy gradient update rule in equation(23):

$$\nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}) = \mathbf{E}_{p_{\boldsymbol{\theta}}(\pi|s)} \left[\nabla \log p_{\boldsymbol{\theta}}(\boldsymbol{\pi} \mid s)(L(\boldsymbol{\pi}) - b(s)) \right] \quad (24)$$

where, $b(s)$ can be considered as a measure of the average performance of the policy. If the current policy's performance is better than this average level $b(s)$, $L(\boldsymbol{\pi}) - b(s)$ is negative, then the policy receives positive reinforcement (minimizing the loss function). Conversely, if the performance of current policy is worse than the average level $b(s)$, $L(\boldsymbol{\pi}) - b(s)$ is positive, then the policy receives negative reinforcement. This section also adopts the greedy rollout baseline method proposed in literature [34] to calculate $b(s)$. The REINFORCE training algorithm uses the Adam optimizer [38] to update parameters θ. The pseudocode for the REINFORCE training process is as follows:

Algorithm 1 Reinforcement learning training method based on REINFORCE

Input: Training Set $\boldsymbol{X}$, batch size $\boldsymbol{B}$, number of epochs N_{epoch}, number of steps per epoch N_{step}

```
Initialize parameters of current and baseline policy θ,  θ* = θ
for epoch = 1 : N_epoch do
    for step = 1 : N_step do
        Randomly generate s_i, i ∈ {1, ..., B} from training set
        Adopt sampling strategy π_i = p_θ (s_i)
        Adopt greedy strategy π_i* = p_θ* (s_i)
        Update the gradient ∇_θ L(θ) using Equation (24).
        Update the parameters θ using the Adam algorithm
    end for
    Validate current policy and baseline policy on validation set
    if p_θ performs better than p_θ*
        θ* = θ
    end if
end for
```

V. VALIDATION AND DISCUSSION

To validate the effectiveness of the PN-attention approach, we tested it in the on-orbit service scenario of multi-target rendezvous. The parameter settings for the ADR task are presented in Table 1. Servicer needs to rendezvous with five targets in orbit to install the TDK device, which can deorbit the decommissioned the targets to graveyard orbit. The PN-attention method proposed in this paper is used for real-time in-orbit sequence planning, and continuous thrust maneuver transfers are performed between the servicing spacecraft and the space targets. The simulation utilizes Differential Evolution (DE) algorithms for comparison to comprehensively validate the effectiveness of the proposed PN-attention method.

TABLE 1. Parameter settings for ADR task.

Parameters	Value
Initial mass of servicer	500 kg
Maximum mass of fuel	350 kg
g_0	9.80665m/ s^2
T_{max}	10N
I_{sp}	300s
μ	3.9860044×10^5 km^3/s^2

A. EXPERIMENT SETUP

Table 2 lists the hyperparameters for model training. For the proposed model in this paper, the following hyperparameters are used: a batch size of 128, hidden layer vector dimensions of 128 for both the encoder and decoder, and the Adam optimizer with a fixed learning rate of 1e−4 for model training. During the REINFORCE training process, 6,400 in-orbit service task planning instances bounded by the orbital range specified in Table 3 were randomly generated as a training set, along with a randomly generated validation set, and the model training was executed for 100 epochs. The orbital elements of the targets of in-orbit service sequence planning were also randomly generated according to the range of orbital elements in Table 3.

TABLE 2. Hyperparameters settings of pointer network.

Parameters	Value
Batch size	128
Number of epochs	100
Instances per epoch	6400
optimizer	Adam
Hidden dimension	128
heads	8
Encoder layers	3
learning rate	1e-4

B. MODEL EVALUATION

All the experiments were conducted in Python. The experimental platform's specifications are: GTX 2070super GPU and Intel Core i7-10700 CPU@ 3.8GHz with 16.0GB of memory.

1) EFFECTIVENESS VALIDATION OF CONTINUOUS THRUST COST ESTIMATOR

This section firstly fives a validation of continuous-thrust cost estimator. The orbital range of training samples for

TABLE 3. Orbital range settings of training database.

Parameters	Minimum value	Maximum value
Orbital altitude (km)	800	1200
Eccentricity	0	0.1
Inclination (deg)	58	62
Right Ascension of Ascending Node(deg)	0	2
Argument of Perigee (deg)	0	360
Mean Anomaly (deg)	0	360
Initial mass of servicer (kg)	300	500

DNN-based continuous-thrust cost estimator is also set in Table 3 and training database set initial mass of servicer as 300-500kg. Data generation utilizes the NPIA [25], which can rapidly solve the Pontryagin's minimum value problem and quickly generate 100,000 continuous-thrust transfer samples in about 3 hours. This transfer sample is used to train the DNN, with hyperparameters set according to the reference article [25]. The following compares the results of DNN-based continuous-thrust cost estimator with the Analytical method [19] and the Lambert estimation method. We selected 1000 samples from dataset for validation.

Figure 6 shows the distribution of estimation errors for the three methods in the continuous-thrust dataset. It is evident that the DNN-based method yields better estimation results, with distributions centered around zero, whereas both the Lambert estimation method and the Analytical method exhibit significant biases, with the Lambert method showing the largest deviation. The DNN-based method approach achieves almost unbiased estimates of continuous-thrust fuel consumption, in contrast to the Lambert and analytical methods, which both have considerable estimation biases.

TABLE 4. Comparison of errors among three continuous thrust cost estimation methods.

Method	Mean error/kg	Standard deviation /kg	Relative error
DNN-based method	2.1452	4.2240	3.986%
Lambert method	25.2809	17.4946	21.729%
Analytical method	12.4798	13.4170	14.064%

Table 4 shows a comparison of the errors of three continuous-thrust cost estimation methods. From the table, it can be seen that the DNN-based method has the smallest average estimated error at 2.1452kg, with a standard deviation of 4.2240kg, indicating a relatively high accuracy of estimation, with an overall estimation error of 3.986%. In contrast, the other two methods have relative errors of 21.729% and 14.064%, respectively, highlighting the relatively high precision of the DNN-based method.

By comparison, it further illustrates the advantages of the DNN-based estimation method. It is evident that the DNN-based method fits complex functional relationships and the trained surrogate model can be used to construct function descriptions that reflect underlying mathematical principles. It can quickly learn and fit complex problems but does not provide an analytical mathematical equation. Compared to the DNN-based approach, the analytical method may lose its ability to fit complex functions due to simplifications in the formulas, missing intrinsic characteristics.

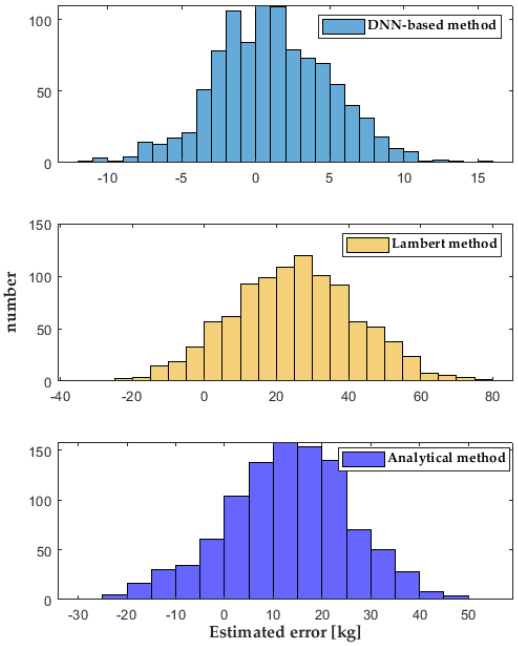

FIGURE 6. Distribution of estimation errors for the three methods on continuous-thrust transfer cost.

2) TRAINING VALIDATION OF PN-ATTENTION FOR ADR PROBLEMS

The following sections will test the training process, solution ability, ablation experiments and generalization performance of PN-attention method.

After training the parameters of PN-attention with the REINFORCE process in each generation, the validation samples were randomly generated to evaluate the loss function. In this process, the PN-attention no longer randomly selects the target based on conditional probability. Instead, it uses a greedy algorithm to choose the action with the highest probability at each step to generate a complete ADR sequence. Set the number of validation samples to 1024 and take the average fuel consumption for all sequence rendezvous as the optimal cost, a smaller of which means better training effect of network parameters.

Average fuel consumption is used as the main index for measuring algorithm performance, which can directly show the training effect of multi-target sequence ADR. The cost convergence curve for ADR mission with five targets is shown in Figure 7 and 8. Figure 7 and 8 display the average convergence curves over ten training sessions.

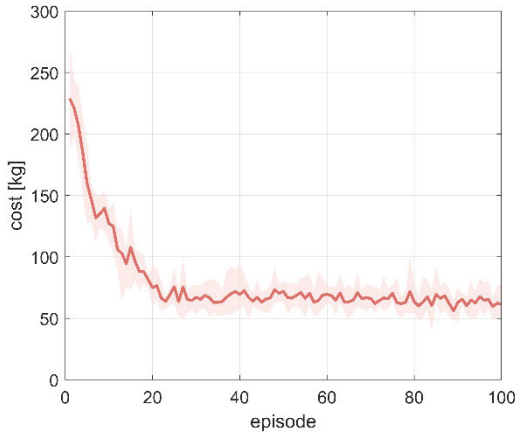

FIGURE 7. Convergence curve of average fuel consumption during training.

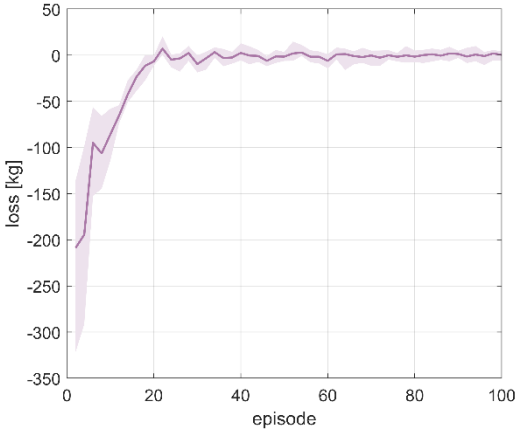

FIGURE 8. Convergence curve of the average loss function during training.

As shown in Figure 7, it can be seen that the average cost of PN-attention model based on the REINFORCE algorithm tends to stabilize after 100 generations of training, with an optimal cost around 70kg. Under this computer configuration, training one generation takes an average of 124s, accumulating to 3.4 hours for 100 generations.

In Figure 8, The loss function of the network gradually converges to 0 as the number of updates increases, indicating that the baseline value is very close to the actual reward value and that the entire training process is becoming stable.

3) ABLATION EXPERIMENT

In order to verify the functions of different structures in PN-attention model, ablation experiment was carried out by deleting or adjusting some structures.

Different model structures are described as follows:

- PN-attention: is the method proposed in this study.
- 4headPN: changes the heads of the attention layer to 4.
- no-attention: deletes the attention mechanism.
- no-baseline: deletes the baseline from the REINFORCE training process

We compared the average fuel consumption of sequences generated by the above four models within the orbital range dataset presented in Table 3, as well as the relative error with the results generated by the DE algorithm, which are shown in Table 5.

From Table 5, it can be seen that the PN-attention method proposed in this paper has the best sequence generation effect in terms of average fuel consumption. To investigate the effect of the attention mechanism on model performance, we reduced the number of attention heads to four (4headPN) and further removed the attention computation (no-attention). It is evident that PN-attention with 8-heads achieves better prediction results and generates optimal average fuel consumption.

TABLE 5. Ablation experiment of different model structures.

Model	Average fuel consumption/kg	Relative error/%
PN-attention	**175.43**	**6.32**
4headPN	206.25	25.12
no-attention	378.54	129.42
no-baseline	221.54	34.27

In order to test the effect of the baseline in the REINFORCE training process, we redesigned the no-baseline model. Similarly, PN-attention shows superior average fuel consumption compared to it and also exhibits significant improvements in training speed.

4) SOLVING CAPABILITY VALIDATION OF PN-ATTENTION FOR ADR PROBLEMS

The following will demonstrate the PN-attention's solution results for sequence planning and compare them with DE algorithms. Figure 9 shows the sequences generated by PN-attention for 100 sets of random ADR sequence planning, with the blue square representing the fuel consumption of PN-attention. For comparison, we introduced the DE algorithm, with the orange circle. Statistically speaking, 90% of the fuel consumption errors are within 30kg, indicating that the sequences generated by PN-attention meet the relatively optimality conditions. However, for rendezvous sequences with small differences in orbital elements, there are many local optima in the total fuel consumption across different time intervals, so there are often multiple solutions. Therefore, such a comparison only has a certain degree of significance.

Table 6 shows the results of three groups of sequences generated by PN-attention. The trajectories planned for all three results are rendezvous with five targets on small-inclination orbits, and there is a significant difference in the fuel consumed. As can be seen from the Table, the result of the first group sequence planning is (1, 2, 5, 4, 3), with a total transfer time of 39Tr and a fuel consumption of 102.096kg. The planned sequences for the other two groups

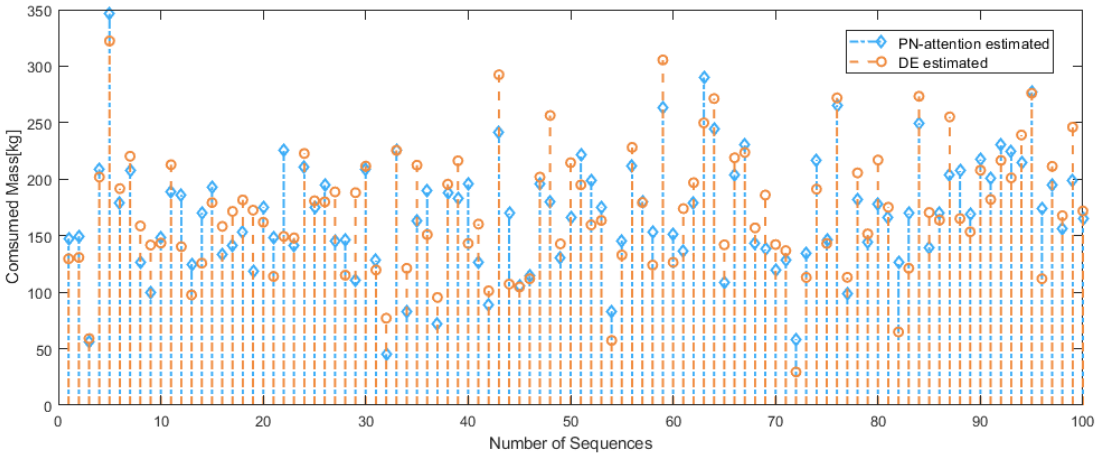

FIGURE 9. Fuel consumption comparison of different algorithms for 100 sequences of random ADR mission.

TABLE 6. Results of three ADR mission.

Group of Sequence	number	Orbital Elements of Targets	Planned Sequence	Transfer Time	Total Cost (kg)
1	#1	[806.1524, 0.0327, 58.1796, 0.8429, 31.2469, 344.1276]	1	2Tr	102.096
	#2	[826.7164, 0.0438, 58.7818, 1.1704, 333.8496, 200.0052]	2	15Tr	
	#3	[1187.372, 0.09593, 61.17936, 1.1378, 5.41116, 89.9928]	5	2Tr	
	#4	[924.0280, 0.0481, 60.2869, 0.5708, 169.5888, 193.1940]	4	12Tr	
	#5	[1120.668, 0.01647, 61.0443, 0.2877, 112.536, 218.2356]	3	8Tr	
2	#6	[915.5600, 0.0632, 61.3072, 0.9536, 33.6600, 11.6280]	6	3Tr	54.840
	#7	[1090.560, 0.02666, 61.7356, 1.1708, 48.852, 244.188]	7	24Tr	
	#8	[1089.2000, 0.00648, 60.3944, 0.9792, 141.804, 306.720]	8	4Tr	
	#9	[946.5600, 0.0117, 58.9748, 1.3504, 275.004, 306.684]	10	6Tr	
	#10	[1184.920, 0.0218, 58.7172, 1.1292, 94.1400, 35.028]	9	3Tr	
3	#11	[1077.360, 0.08273, 60.8192, 1.4154, 74.016, 92.664]	13	3Tr	160.632
	#12	[916.320, 0.045, 61.2584, 1.3776, 193.2480, 304.308]	11	30Tr	
	#13	[1057.040, 0.00862, 61.9728, 0.3862, 315.684, 221.184]	15	18Tr	
	#14	[828.760, 0.0370, 61.8416, 1.2244, 217.1520, 283.068]	14	10Tr	
	#15	[1052.640, 0.03424, 59.5004, 1.0912, 231.228, 209.556]	12	16Tr	

*Tr = 1000s

are (6, 7, 8, 10, 9) and (13, 11, 15, 14, 12). The second group consumed the least fuel, using a total of 54.840kg and taking the least time of 40Tr, while the most was 160.632kg, taking 77Tr of group 3. PN-attention will only provide a unique sequence output based on the input, which has a high possibility of falling into a local optimal solution. Therefore, after PN-attention provides a sequence, further optimization using a local optimization algorithm can yield more reasonable results [34].

5) GENERALIZATION VALIDATION

The comparison results of the speed and optimality of multiple algorithms are shown in the Table 7. Here, the comparisons are made with different algorithms: DE Algorithm, Upper Confidence bound Tree (UCT) search algorithm, Ant Colony Optimization (ACO), and Particle Swarm Optimization (PSO). When generating 100 sequences, we compared the total computation time, average fuel consumption, and average transfer time among the different algorithms. In the comparative experiment, PN-attention is still trained for the 5-Target ADR problem.

First, we compare the solution results on the 5-Target ADR problem. In terms of total computation time, PN-attention requires only 13.5 seconds, UCT-ADR takes 1725.6 seconds, while the other algorithms all need around ten thousand seconds. It is evident that PN-attention's solution speed is

TABLE 7. Generalization performance comparison of different algorithms on ADR problems of generating 100 sequences with different numbers of targets.

Method	3-Target ADR			5-Target ADR			7-Target ADR		
	TCT/s	AFC /kg	ATT /Tr	TCT/s	AFC /kg	ATT /Tr	TCT/s	AFC /kg	ATT /Tr
PN-attention	6.2	109.57	86.25	13.5	178.42	154.35	27.4	221.45	190.43
DE	3674.2	102.27	88.54	10692.2	169.35	126.34	15672.4	215.47	184.52
UCT-ADR[29]	652.4	136.52	102.39	1725.6	182.06	145.27	2263.1	287.03	202.69
ACO[39]	1263.5	105.09	85.01	8976.1	167.92	134.85	13629.5	202.45	191.37
PSO[40]	4836.0	127.45	90.23	12809.8	175.43	189.76	18643.2	256.43	196.54

* TCT = Total Computational Time, AFC = Average fuel consumption, ATT = Average transfer time, Tr = 1000s

nearly 800 times faster than the DE algorithm. In terms of sequence optimality, the average fuel consumption for the 100 sequences generated by PN-attention is 178.42kg, whereas the heuristic DE algorithm is at 169.35kg, ACO is at 167.92kg, and PSO is at 175.43kg. PN-attention has a relative error of 5.36% compared to the DE algorithm, better than UCT-ADR, which has the same reinforcement learning training. As for the average transfer time, the five algorithms have respective times of 154.35, 126.34, 145.27, 134.85, and 189.76 seconds, showing variances. In summary, the PN-attention proposed in this paper ensures a fast sequence generation speed while also meeting relatively good optimality during the sequence generation process.

Next, we tested the generalization capability of the PN-attention method by applying it to scenarios involving 3 and 7 targets for multi-target rendezvous sequence generation. We used the trained PN-attention model for 5-target planning to generate sequence for 100 random scenarios in both 3-target and 7-target missions, and compared the results with those obtained from the other methods.

The first and third columns of Table 7 give the results of different algorithms on 3-Target ADR and 7-Target ADR, respectively. In the 3-Target ADR task scenario, DE provided an optimal solution with an average of 102.27kg. The AFC generated by PN-attention was 109.57kg, with a relative error of 6.67% compared to the optimal solution. Among the five methods, PN-attention has the fastest sequence generation speed at 6.2 seconds, which is 100 times faster than the UCT-ADR method. In the 7-Target ADR task scenario, ACO provided an optimal solution with an average of 202.45kg. The AFC for PN-attention was 221.45kg, with a relative error of 8.58% compared to the optimal solution. Similarly, in the 7-Target ADR problem, the PN-attention method still maintains the highest solution efficiency. From above comparisons, it can be seen that PN-attention, while ensuring a high solution speed when solving ADR problems, can maintain a certain level of generalization performance.

VI. CONCLUSION

In this study, we proposed a PN-attention method for ADR problems that integrates an attention mechanism with a pointer network. A DNN network was trained to enable the rapid estimation of continuous-thrust costs in ADR processes. PN-attention is introduced, which can dynamically adjust its strategy according to the current state of targets to optimize the sequence generation process. Based on the PN-attention model, this research extends the action space by discretizing continuous time and combining target orders, allowing the attention model to select different targets at different time points. Simulation experiments were conducted to test the performance of the effectiveness, optimality, and generalization of the PN-attention. In particular, compared to heuristic algorithms, PN-attention achieves a 100-times improvement in computational speed while ensuring optimality.

REFERENCES

[1] D. J. Kessler and B. G. Cour-Palais, "Collision frequency of artificial satellites: The creation of a debris belt," *J. Geophys. Res., Space Phys.*, vol. 83, no. A6, pp. 2637–2646, Jun. 1978, doi: 10.1029/ja083ia06p02637.

[2] D. Madakat, J. Morio, and D. Vanderpooten, "Biobjective planning of an active debris removal mission," *Acta Astronautica*, vol. 84, pp. 182–188, Mar. 2013, doi: 10.1016/j.actaastro.2012.10.038.

[3] D. Izzo, I. Getzner, D. Hennes, and L. F. Simões, "Evolving solutions to TSP variants for active space debris removal," in *Proc. Annu. Conf. Genetic Evol. Comput.*, Jul. 2015, pp. 1207–1214, doi: 10.1145/2739480.2754727.

[4] M. Cerf, "Multiple space debris collecting mission: Optimal mission planning," *J. Optim. Theory Appl.*, vol. 167, no. 1, pp. 195–218, Oct. 2015, doi: 10.1007/s10957-015-0705-0.

[5] Y. Liu, J. Yang, Y. Wang, Q. Pan, and J. Yuan, "Multi-objective optimal preliminary planning of multi-debris active removal mission in LEO," *Sci. China Inf. Sci.*, vol. 60, no. 7, Jul. 2017, Art. no. 7, doi: 10.1007/s11432-016-0566-7.

[6] J. Guo, Z. Pang, and Z. Du, "Optimal planning for a multi-debris active removal mission with a partial debris capture strategy," *Chin. J. Aeronaut.*, vol. 36, no. 6, pp. 256–265, Jun. 2023, doi: 10.1016/j.cja.2023.03.013.

[7] J. Bang and J. Ahn, "Multitarget rendezvous for active debris removal using multiple spacecraft," *J. Spacecraft Rockets*, vol. 56, no. 4, pp. 1237–1247, Jul. 2019, doi: 10.2514/1.a34344.

[8] H. Li, S. Chen, D. Izzo, and H. Baoyin, "Deep networks as approximators of optimal low-thrust and multi-impulse cost in multitarget missions," *Acta Astronautica*, vol. 166, pp. 469–481, Jan. 2020, doi: 10.1016/j.actaastro.2019.09.023.

[9] G. Gatto and L. Casalino, "Fast evaluation and optimization of low-thrust transfers to multiple targets," *J. Guid., Control, Dyn.*, vol. 38, no. 8, pp. 1525–1530, Aug. 2015, doi: 10.2514/1.g001116.

[10] M. Di Carlo, J. M. Romero Martin, and M. Vasile, "Automatic trajectory planning for low-thrust active removal mission in low-Earth orbit," *Adv. Space Res.*, vol. 59, no. 5, pp. 1234–1258, Mar. 2017, doi: 10.1016/j.asr.2016.11.033.

[11] R. Chen, Y. Bai, Y. Zhao, Y. Wang, W. Yao, and X. Chen, "Closed-loop optimal control based on two-phase pseudospectral convex optimization method for swarm system," *Aerosp. Sci. Technol.*, vol. 143, Dec. 2023, Art. no. 108704, doi: 10.1016/j.ast.2023.108704.

[12] T. J. Betts, "Practical methods for optimal control and estimation using nonlinear programming," *Appl. Mech. Rev.*, vol. 55, no. 4, pp. 1–13, 2010.

[13] J. A. Kechichian, "Mechanics of trajectory optimization using nonsingular variational equations in polar coordinates," *J. Guid., Control, Dyn.*, vol. 20, no. 4, pp. 812–818, Jul. 1997, doi: 10.2514/2.4117.

[14] C. L. Ranieri and C. A. Ocampo, "Indirect optimization of spiral trajectories," *J. Guid., Control, Dyn.*, vol. 29, no. 6, pp. 1360–1366, Nov. 2006, doi: 10.2514/1.19539.

[15] M. La Mantia and L. Casalino, "Indirect optimization of low-thrust capture trajectories," *J. Guid., Control, Dyn.*, vol. 29, no. 4, pp. 1011–1014, Jul. 2006, doi: 10.2514/1.18986.

[16] J. T. Betts, "Survey of numerical methods for trajectory optimization," *J. Guid., Control, Dyn.*, vol. 21, no. 2, pp. 193–207, Mar. 1998, doi: 10.2514/2.4231.

[17] F. Jiang, H. Baoyin, and J. Li, "Practical techniques for low-thrust trajectory optimization with homotopic approach," *J. Guid., Control, Dyn.*, vol. 35, no. 1, pp. 245–258, Jan. 2012, doi: 10.2514/1.52476.

[18] F. Zuiani and M. Vasile, "Preliminary design of debris removal missions by means of simplified models for low-thrust, many-revolution transfers," *Int. J. Aerosp. Eng.*, vol. 2012, no. 1, pp. 1–22, 2012, doi: 10.1155/2012/836250.

[19] H. Li, S. Chen, and H. Baoyin, "J2-perturbed multitarget rendezvous optimization with low thrust," *J. Guid., Control, Dyn.*, vol. 41, no. 3, pp. 802–808, Mar. 2018, doi: 10.2514/1.g002889.

[20] M. K. Jorgensen and I. Sharf, "Optimal planning for a multiple space debris removal mission using high-accuracy low-thrust transfers," *Acta Astronautica*, vol. 172, pp. 56–69, Jul. 2020, doi: 10.1016/j.actaastro.2020.03.031.

[21] S. Narayanaswamy, B. Wu, P. Ludivig, F. Soboczenski, K. Venkataramani, and C. J. Damaren, "Low-thrust rendezvous trajectory generation for multi-target active space debris removal using the RQ-law," *Adv. Space Res.*, vol. 71, no. 10, pp. 4276–4287, May 2023, doi: 10.1016/j.asr.2022.12.049.

[22] H. Shen, J. Zhou, Q. Peng, Y. Luo, and H. Li, "Low-thrust trajectory optimization for multiple target bodies tour mission," *Theor. Appl. Mech. Lett.*, vol. 1, no. 5, 2011, Art. no. 053001, doi: 10.1063/2.1105301.

[23] D. Zona, A. Zavoli, L. Federici, and G. Avanzini, "Evolutionary optimization for active debris removal mission planning," *IEEE Access*, vol. 11, pp. 41019–41033, 2023, doi: 10.1109/ACCESS.2023.3269305.

[24] L. Böttcher, N. Antulov-Fantulin, and T. Asikis, "AI pontryagin or how artificial neural networks learn to control dynamical systems," *Nature Commun.*, vol. 13, no. 1, p. 9, Jan. 2022, doi: 10.1038/s41467-021-27590-0.

[25] Z. Chen, J. Luo, Q. Chen, Y. Zhao, Y. Bai, and X. Chen, "Fast estimation of initial costate for time-optimal trajectory based on surrogate model," *J. Aerosp. Eng.*, vol. 36, no. 6, Nov. 2023, Art. no. 04023078, doi: 10.1061/jaeeez.aseng-4876.

[26] G. Viavattene, E. Devereux, D. Snelling, N. Payne, S. Wokes, and M. Ceriotti, "Design of multiple space debris removal missions using machine learning," *Acta Astronautica*, vol. 193, pp. 277–286, Apr. 2022, doi: 10.1016/j.actaastro.2021.12.051.

[27] X. Guo, D. Ren, D. Wu, and F. Jiang, "DNN estimation of low-thrust transfer time: Focusing on fast transfers in multi-asteroid rendezvous missions," *Acta Astronautica*, vol. 204, pp. 518–530, Mar. 2023, doi: 10.1016/j.actaastro.2022.09.006.

[28] Y. Xu, X. Liu, R. He, Y. Zhu, Y. Zuo, and L. He, "Active debris removal mission planning method based on machine learning," *Mathematics*, vol. 11, no. 6, p. 1419, Mar. 2023, doi: 10.3390/math11061419.

[29] J. Yang, X. Hou, Y. H. Hu, Y. Liu, and Q. Pan, "A reinforcement learning scheme for active multi-debris removal mission planning with modified upper confidence bound tree search," *IEEE Access*, vol. 8, pp. 108461–108473, 2020, doi: 10.1109/ACCESS.2020.3001311.

[30] J. Perera, S.-H. Liu, M. Mernik, M. Črepinšek, and M. Ravber, "A graph pointer network-based multi-objective deep reinforcement learning algorithm for solving the traveling salesman problem," *Mathematics*, vol. 11, no. 2, p. 437, Jan. 2023, doi: 10.3390/math11020437.

[31] O. Vinyals, M. Fortunato, and N. Jaitly, "Pointer networks," 2015, *arXiv:1506.03134*.

[32] A. Vaswani, "Attention is all you need," Aug. 2023, *arXiv:1706.03762*.

[33] I. Bello, H. Pham, Q. V. Le, M. Norouzi, and S. Bengio, "Neural combinatorial optimization with reinforcement learning," 2016, *arXiv:1611.09940*.

[34] W. Kool, H. van Hoof, and M. Welling, "Attention, learn to solve routing problems!" 2018, *arXiv:1803.08475*.

[35] M. M. Castronuovo, "Active space debris removal—A preliminary mission analysis and design," *Acta Astronautica*, vol. 69, nos. 9–10, pp. 848–859, Nov. 2011, doi: 10.1016/j.actaastro.2011.04.017.

[36] D. Izzo and E. Öztürk, "Real-time guidance for low-thrust transfers using deep neural networks," *J. Guid., Control, Dyn.*, vol. 44, no. 2, pp. 315–327, Feb. 2021, doi: 10.2514/1.g005254.

[37] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Mach. Learn.*, vol. 8, nos. 3–4, pp. 229–256, May 1992, doi: 10.1007/bf00992696.

[38] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*.

[39] T. Zhang, H. Shen, Y. Yang, H. Li, and J. Li, "Ant colony optimization-based design of multiple-target active debris removal mission," *Trans. Jpn. Soc. Aeronaut. SPACE Sci.*, vol. 61, no. 5, pp. 201–210, 2018, doi: 10.2322/tjsass.61.201.

[40] J. Yu, X.-Q. Chen, and L.-H. Chen, "Optimal planning of LEO active debris removal based on hybrid optimal control theory," *Adv. Space Res.*, vol. 55, no. 11, pp. 2628–2640, Jun. 2015, doi: 10.1016/j.asr.2015.02.026.

ZHIJUN CHEN received the M.Eng. degree from the National University of Defense Technology, Changsha, China, in 2019, where he is currently pursuing the Ph.D. degree. His research interests include guidance and control, optimal control, intelligent control, and trajectory optimization.

YUZHU BAI received the M.Sc. and Ph.D. degrees in aerospace engineering from the National University of Defense Technology, Changsha, China, in 2005 and 2011, respectively. He is currently an Associate Professor with the National University of Defense Technology. His current research interests include spacecraft orbital dynamics and control, and micro-nano satellite design.

YONG ZHAO received the Ph.D. degree in aerospace science and engineering from the National University of Defense Technology, Changsha, China, in 2005. He is currently a Professor with the National University of Defense Technology. His research interests include micro-nano satellite design, spacecraft cluster application, and topological optimization.

XIAOQIAN CHEN received the M.S. and Ph.D. degrees in aerospace engineering from the National University of Defense Technology, China, in 1997 and 2001, respectively. He is currently a Researcher of Chinese Academy of Military Science, Beijing, China. His current research interests include spacecraft systems engineering, advanced digital design methods of space systems, and multidisciplinary design optimization.