Hindawi
Wireless Communications and Mobile Computing
Volume 2022, Article ID 5265038, 14 pages
https://doi.org/10.1155/2022/5265038

Research Article

A Novel Path Planning and Node Selection Method Using Reinforcement Learning in NTN IoT Networks

Siming Yang[1] **Zheng Shan,**[1] **Jiang Cao,**[2] **Yuan Gao,**[2] **Yang Guo,**[2] **Ping Wang,**[2] **Jing Wang,**[2] **and Xiaonan Wang**[2]

[1]*State Key Laboratory of Mathematical Engineering and Advanced Computing, Zhengzhou, Henan 450001, China*
[2]*Academy of Military Science of the PLA, 100091, China*

Correspondence should be addressed to Siming Yang; 972856350@qq.com

Received 16 May 2022; Accepted 25 August 2022; Published 16 September 2022

Academic Editor: Mingqian Liu

With the rapid deployment of 5G networks in recent years, the characteristics of high bandwidth, low latency, and low energy consumption of 5G networks have enabled the rapid development of IoT (Internet of things) technology. However, 5G networks cannot provide high-quality wireless coverage for many IoT devices in border areas and hotspots with a high signal density that lack fixed infrastructure. Therefore, this paper uses the UAV (unmanned aerial vehicle) to carry the communication platform to build the NTN (nonterrestrial network) to provide wireless coverage for terrestrial fixed and mobile IoT devices. Meanwhile, since the NTN needs to provide wireless coverage for many IoT devices, we use deep reinforcement learning to provide path planning for the UAV communication platform to improve the efficiency of wireless coverage. We build a simulation environment to evaluate the performance of the NTN network for wireless coverage of IoT devices in urban hotspot areas. Experimental results show that the method proposed in this paper can provide higher downlink rates for more IoT devices than NB-IoT (narrowband Internet of things).

1. Introduction

In recent years, IoT technology has made great progress with the advent of 5G networks. Stable, high-quality, and wide wireless coverage are necessary prerequisite for the development and application of IoT technology. The widely used NB-IoT (narrowband Internet of things) [1] technology is mainly used in applications with small data volumes and low rates. The advantage of NB-IoT is lower power consumption and cost. At present, with the rise of video and audio applications, the low rate of NB-IoT technology limits the development of the IoT. The high-speed and low-latency characteristics of the 5G network make up for the low transmission rate. On the other hand, the massive MIMO (multiple input multiple output) technology proposed by the 5G network utilizes the spatial independence of users to provide independent narrow-beam coverage for different users in space and simultaneously transmit data of different users to improve system throughput [2]. The generation of this technology makes it possible to increase the capacity of the system while ensuring a high transmission rate. However, 5G networks cannot provide high-quality wireless coverage in border areas and hotspots with a high signal density and that lack fixed infrastructure. The IoT has two characteristics. One is that the number of connected devices is large, and the other is that the services of IoT devices are mainly burst services. Therefore, how to reasonably provide high-quality wireless coverage for a large number of IoT devices is a crucial problem to be solved at present. In order to promote the development and application of IoT in border areas and hotspot areas, many countries are now discussing the realization of high-quality signal coverage for IoT through NTN (nonterrestrial networks) [3] under the 6G standard.

Compared with traditional terrestrial base station networking, NTN uses HAP (high-altitude platform) for networking. Taking advantage of the maneuverability of HAP and the characteristics of line-of-sight communication, NTN can provide high-quality signal coverage for border areas without ground base station coverage and disaster areas where ground base stations are damaged due to disasters. At the same time, in urban hot spots, NTN can assist ground base stations in solving the problem of wireless network congestion. The current challenge is that the NTN communication platform needs to serve many IoT devices and the distance from the NTN platform to the terminal is significantly larger than that of the ground base station. In addition, the Doppler frequency shift caused by the relative motion between the NTN platform and the IoT device can lead to fast fading. Therefore, avoiding fast fading while enabling the NTN platform to efficiently provide wireless coverage for terrestrial fixed and mobile IoT devices is an urgent problem to be solved.

This paper adopts a UAV (unmanned aerial vehicle) equipped with a base station called UAV-AP (unmanned aerial vehicle-aerial platform) as the NTN high-altitude platform to provide wireless coverage for urban IoT devices. We design a path planning algorithm based on deep reinforcement learning for the NTN communication platform. The algorithm enables the NTN platform to adjust its flight trajectory in real time according to the terrain characteristics and the IoT device's location information. With the support of algorithms, the NTN platform can provide services for as many IoT devices as possible and maximize the sum of the downlink rates of ground IoT devices while avoiding rapid fading.

The path planning method based on deep reinforcement learning has been widely used in scenarios where a UAV is used as a base station platform. Guo et al. and Bayerlein et al. [4, 5] used DQN [6] to solve the trajectory optimization problem to maximize the communication rate of transmission. However, DQN can only be applied to tasks in discrete action spaces and have a defect of overestimating the value function. Wang et al. [7] studied the optimal deployment of UAVs to maximize the real-time downlink capacity by using the double DQN algorithm [8]. Double DQN revises the defect of overestimation, but it is still unable to control the agent's actions in the continuous action space task. Liu et al. and Qi et al. [9, 10] adopted the DDPG algorithm [11] to maximize the geographical fairness of all considered points of interest (PoIs) and minimized the total energy consumptions. The DDPG (deep deterministic policy gradient) algorithm has successfully applied the algorithm to continuous action space tasks. However, the DDPG algorithm has too many hyperparameters that need to be adjusted, leading to slow and unstable agent training.

The main contributions of this paper are as follows:

(1) This paper proposes a UAV-AP air-to-ground urban propagation model. Meanwhile, we summarize the task as an optimization problem by combining the air-to-ground channel model and requirements of the UAV-AP communication support task

(2) We built a simulation environment that meets the task requirements based on the OpenAI Gym environment. The simulation environment reserves an interface for the interaction of reinforcement learning algorithms, which can efficiently train the agent

(3) This paper proposes an intrinsic reward reinforcement learning algorithm based on a parallel architecture. The algorithm can not only be used for policy optimization in continuous action space but also has a higher learning speed and better performance than previous reinforcement learning algorithms. The algorithm enables the NTN platform to perform real-time path planning based on multiple static and mobile IoT devices in a simulated environment, efficiently providing wireless coverage for IoT devices

The rest of this paper is organized as follows. Section 2 describes the NTN IoT network and generalizes the wireless coverage task as an optimization problem. In Section 3, we give a brief introduction to deep reinforcement learning and provide an analysis of our proposed algorithm. Then, we built a simulation environment, carried out related experiments, and analyzed the experimental results. Finally, Section 5 concludes the paper and looks forward to future work.

2. Problem Formulation

In this chapter, we will analyze the task of the NTN communication platform to provide wireless coverage for terrestrial IoT devices. Then, we introduce an air-to-ground channel model to estimate the path loss between NTN platforms and mobile IoT devices in urban environments. At the same time, we define the symbols used in subsequent articles and obtain the mathematical expression of the NTN platform signal coverage task.

The use of the NTN communication platform efficiently protects the burst services of a large number of static or mobile IoT devices on the ground. For example, in the rescue operation of the Henan flood in 2021, the NTN communication platform has achieved long-term stable continuous mobile signal coverage in the disaster area of about 50 square kilometers. The NTN communication platform has connected 2,572 users, generating 1,089.89 M of traffic, with a maximum of 648 users accessing a single time. The NTN communication platform restored the mobile public network signal for residents in the disaster area to promptly report the disaster situation and location information and also provided communication support for the rescue team.

As shown in Figure 1, the NTN communication platform can rely on its mobility to efficiently provide wireless connections for burst communications in multiple areas.

When providing wireless coverage for IoT devices in the target area, the NTN platform needs to comprehensively consider the signal attenuation caused by the building and the location information of the IoT device to adjust its flight trajectory in real time. The NTN platform needs to ensure that the downlink rate of all IoT devices is higher than the minimum threshold rate and avoid fast fading caused by relative motion.

2.1. Modeling of the Air-to-Ground Channel. We introduce an air-to-ground channel path loss model based on the

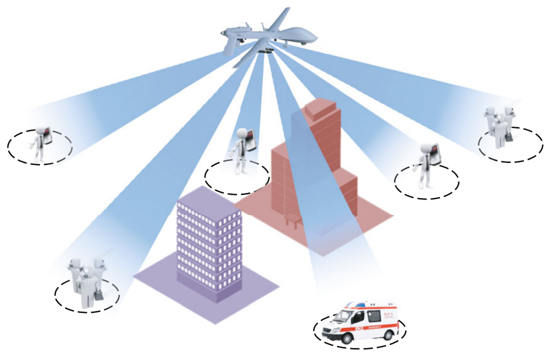
FIGURE 1: Schematic diagram of the NTN network wireless coverage task.

urban environment, mainly considering the path loss caused by the signal blocking by urban buildings. The International Telecommunication Union (ITU) proposes a general model based on the loss of radio signal transmission caused by the occlusion of buildings in its official standard. The model can be applied to a variety of urban environments. The transmission probability of line-of-sight communication and non-line-of-sight communication between the transmitter and the receiver is defined as a function of elevation angle and environmental parameters. Through mathematical derivation, the formula is achieved, which is gradually evolved and simplified by SIGMOD.

The International Telecommunication Union (ITU) [12] proposed an air-to-ground channel model that can be used in a variety of urban environments to measure the geometric probability of LoS transmission between the transmitter and the receiver. This model summarizes the probability of LoS (line of sight) and NLoS (nonline of sight) as a function of elevation angle and several environmental parameters. The formula can be fitted by the SIGMOD function which can be written as follows:

$$\mathbf{P}(\mathbf{LoS},\theta) = \frac{1}{1 + a \exp\left(-b[\theta - a]\right)}. \tag{1}$$

Parameters "a" and "b" are called S-curve parameters, which can be queried in literature [12] according to the corresponding city type.

According to the obtained LoS probability, the NLoS probability is written as follows:

$$\mathbf{P}(\mathbf{NLoS},\theta) = 1 - \mathbf{P}(\mathbf{LoS},\theta). \tag{2}$$

Therefore, the path loss of the propagation can be modeled as follows:

$$\mathbf{PL}_{\xi} = \mathbf{FSPL} + \eta_{\xi}, \tag{3}$$

where **FSPL** represents the free space pathloss [13] between the NTN platform and an IoT device. η_{ξ} is the excessive path loss which is determined by the environment, and ξ refers to the propagation group. In this paper, we divide the propagation model into LoS and NLoS, which means that $\xi \in \{\text{LoS}, \text{NLoS}\}$, and the total path loss can be modeled as the following:

$$\mathbf{PL} = \mathbf{P}(\mathbf{LoS},\theta) \times \mathbf{PL}_{\text{LoS}} + \mathbf{P}(\mathbf{NLoS},\theta) \times \mathbf{PL}_{\text{NLoS}}, \tag{4}$$

where PL is the total path loss of the channel model and the downlink rate of each device can be calculated according to the path fading formula. Using the general model, we can obtain the path loss values of the UAV and the ground target at different elevation angles under the premise of free space loss. The general path loss calculation method for air-to-ground channels is the basis for generalizing the UAV-AP communication coverage task as an optimization problem.

2.2. Mathematical Expression of the Problem. We define the signal path loss from the NTN platform to an IoT device $m (m \in M)$ as PL_m, speed of light as c, signal frequency as f, NTN platform's base station transmit power as P_s, the bandwidth as W, and the position vector from the NTN platform to the IoT device m as $\vec{d}_m$. Besides that, we set the velocity vectors of the NTN platform and an IoT device to be $\vec{v}_f$ and $\vec{v}_m$, respectively. The Gaussian white noise power on the bandwidth is set to N, and the minimum threshold rate of IoT device m is D_m. Figure 2 is a schematic diagram of the relationship between the NTN platform and a mobile IoT device.

The frequency of the signal received by the device m at time t can be written as follows:

$$f_{mt} = f \times \left(\frac{c - \vec{v}_m \cdot \vec{d} / \left|\vec{d}\right|}{c - \vec{v}_f \cdot \vec{d} / |d|} \right). \tag{5}$$

The IoT device m that received power at time t is shown as follows:

$$\mathbf{Pr}_{\mathbf{mt}} = 10 \lg \left(1000 \times P_s\right) - PL. \tag{6}$$

We can get the theoretical maximum downlink rate from Shannon's formula [13], which can be written in the following form:

$$\mathbf{C}_{\mathbf{mt}} = W \log_2 \left(1 + \frac{\mathbf{Pr}_{\mathbf{mt}}}{N}\right), \tag{7}$$

where $\mathbf{C}_{\mathbf{mt}}$ is the downlink rate of the device m at time t. The NTN communication platform continues to provide wireless coverage for terrestrial IoT devices during time T. During this period, the time for device m to perform burst service is T_m. Under the condition that each IoT device does not produce fast fading and the downlink rate is higher than the threshold rate, our goal is to maximize the average downlink rate of M devices under the effective communication time in a period T. In time T, we set the time set of user

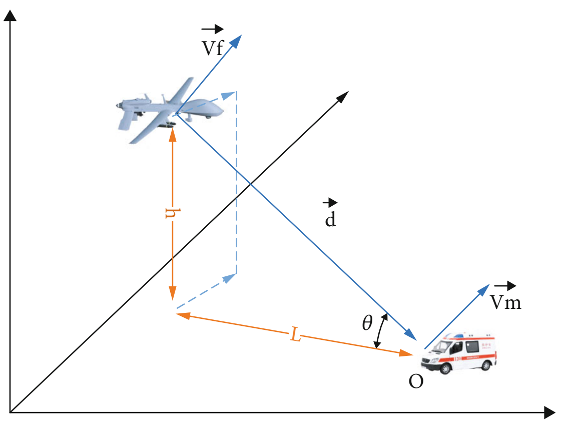

FIGURE 2: Relative position relationship between an NTN platform and a ground mobile IoT device in a certain time slot.

m to perform burst service as G_m. In order to avoid the fast fading of IoT devices, it is necessary to ensure that the symbol time is longer than the coherence time when the device performs burst communication, which means that when $t \in G_m$, $C_{mt} f_{mt}$. So, our problem can be summarized as the following optimization problem:

$$\begin{aligned} &\text{maximum} \frac{1}{M} \times \sum_{m}^{M} \left(\frac{1}{T_m} \times \sum_{t=0}^{T} C_{mt} \right) \\ &\text{subject to } C_{mt} f_{mt}, \quad \text{when } t \in G_m, \\ &C_{mt} \geq D_m, \quad \text{when } t \in G_m. \end{aligned} \tag{8}$$

It can be seen that the UAV-AP wireless coverage task is an optimization problem with many unknowns. Such problems are difficult to solve using optimization theory, so we use deep reinforcement learning to optimize the policy in the interaction between the agent and the environment. The optimization objective in equation (8) is the expected return of the DRL algorithm, and the constraints limit the form of the algorithm's reward function.

3. Algorithm Analysis

From the conclusions of Section 2, we can see that the task of an NTN communication platform to provide wireless coverage for terrestrial multi-IoT devices involves many variables. It is difficult to find the optimal solution with traditional optimization methods for this problem. Therefore, we intend to use reinforcement learning methods to improve the policy using the experience generated by the agent (i.e., the NTN communication platform) interacting with the environment. Eventually, the agent can learn a near-optimal policy that satisfies our goal.

In this section, we first introduce deep reinforcement learning and then propose our improved method for the slow learning speed of the current algorithms.

3.1. Introduction of DRL. RL (reinforcement learning) [14] originated from the optimal control theory in cybernetics. It is mainly used to solve sequential decision-making problems. Through continuous interaction and trial and error with the environment, the agent finally learns a near-optimal strategy and maximizes the expected revenue. DRL (deep reinforcement learning) is a combination of DL (deep learning) [15] and reinforcement learning. With the help of deep neural networks' powerful feature representation capabilities, it can fit various functions in reinforcement learning well, enhance the fitting ability of reinforcement learning, and expand the RL application scenarios.

Our research sets the NTN communication platform as the agent and sets up a simulated environment to interact with the agent. Using the empirical data generated by the interaction between the agent and the environment, the agent can finally learn a near-optimal path planning strategy. The NTN communication platform makes real-time decisions according to the strategy obtained by training, maximizing the average downlink rate of the ground IoT devices under adequate communication time.

A drawback of the current DRL is that the algorithm learns slowly, especially when dealing with tasks with ample state space and action space. Therefore, we need to design an algorithm that can efficiently process a large amount of empirical data and rapidly improve the strategy's performance. The training process of reinforcement learning can be divided into the empirical data collection stage and the agent training stage. The former is responsible for collecting interaction data between the agent and the environment, and the latter is responsible for training strategies using the collected data. When we discuss the acceleration of the algorithm, both stages need to be considered in detail. Methods to improve the learning speed of the agent refer to Figure 3.

In the empirical data collection stage, the primary methods to accelerate the learning rate are to improve the quality of empirical data or improve the efficiency of data collection. In order to improve the sample quality, we can use the algorithms which can guide the agent to explore the unknown environment efficiently, thereby increasing the diversity of the collected data. In order to improve the efficiency of data collection, the current mainstream method is to combine reinforcement learning algorithms with parallel architectures.

The main methods to improve the learning speed in the strategy training phase can be summarized as follows. The first type of method balances the deviation and variance of the value function estimation. The second type of method is to utilize the information of value distribution. The third type of method is to stabilize the training process of the agent and avoid excessive fluctuations that affect the convergence speed and results. The fourth method is to improve data utilization efficiency, which can replay critical empirical data to accelerate algorithm convergence.

3.2. Algorithm. In order to improve the learning speed of the algorithm, we refer to the RND algorithm [17] and introduce intrinsic rewards in order that the agent can make the training process stable and improve exploration efficiency to obtain high-quality data. RND innovatively uses intrinsic reward to measure the novelty of the current state,

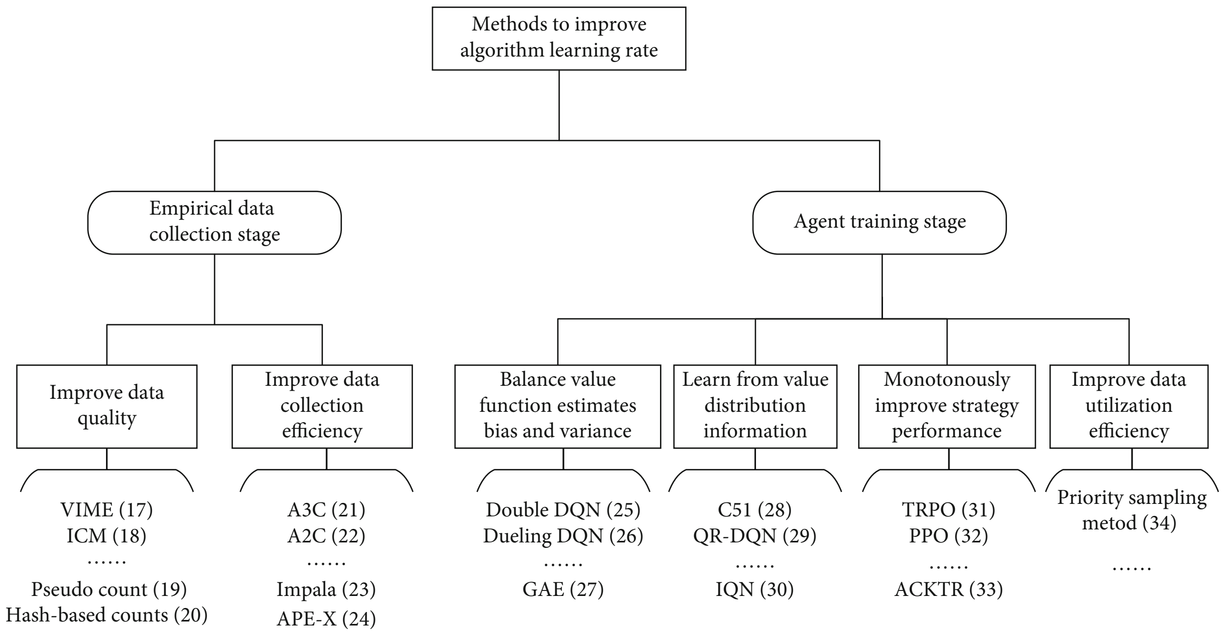
Methods to improve algorithm learning rate
Empirical data collection stage
Agent training stage
Improve data quality
Improve data collection efficiency
Balance value function estimates bias and variance
Learn from value distribution information
Monotonously improve strategy performance
Improve data utilization efficiency
VIME (17)
ICM (18)
......
Pseudo count (19)
Hash-based counts (20)
A3C (21)
A2C (22)
......
Impala (23)
APE-X (24)
Double DQN (25)
Dueling DQN (26)
......
GAE (27)
C51 (28)
QR-DQN (29)
......
IQN (30)
TRPO (31)
PPO (32)
......
ACKTR (33)
Priority sampling metod (34)
......

Figure 3: Methods to improve algorithm learning speed.

which encourages the agent to explore the strange state action combination of the simulation environment. In addition, the RND algorithm incorporates the idea of the PPO [16] algorithm and achieves a monotonous increase in agent performance during the training process, which improves the stability of the training process. Based on the RND algorithm, our proposed algorithm integrates it into a parallel architecture to accelerate the learning rate of the algorithm by improving the efficiency of data collection.

Next, we briefly introduce the adopted architecture and then propose solutions for the problem of bias caused by strategy learning.

3.2.1. Architecture Introduction. The architecture of the RND algorithm is summarized in Figure 4. The problem of the algorithm is that the data collection efficiency is too low.

In order to further improve the data sampling efficiency, we refer to the idea of Impala architecture. The Impala architecture contains multiple workers and one or more learners. The worker is responsible for interacting with the independent environment and collecting experience data. The learner is responsible for updating the strategy with aggregated data. Figure 5 shows the architecture diagram with multiple workers and one learner.

Each worker is independent. In Figure 5, worker i completes an episode of data interaction, stores the collected experience data in the cache, then synchronizes the current latest strategy from the learner, and starts a new round of interaction. The learner will use the collected data from multiepisode to update the strategy. After completing an episode of interaction and the experience data is saved in the cache, worker i will synchronize the latest strategy of the learner instead of waiting for the strategy update of the learner. This architecture completely decouples the learner and the worker, saves the worker's waiting time, and dramatically accelerates the speed of data collection. Figure 6 shows the timeline comparison between the learner and the worker in Impala, A3C, and A2C algorithms.

It is shown that each worker of the A3C architecture needs to wait for the strategy update of the learner to synchronize its latest strategy and uses the new policy to start the next episode of interaction. In A2C architecture, the learner needs to wait for all workers to complete the interaction task and uses data to update policy and then synchronize policy to all workers. Compared to the above architectures, workers and learners of the Impala architecture are entirely decoupled. With the simple architecture, the learner can conveniently use the GPU for acceleration and the workers can be easily distributed on many machines.

3.2.2. Intrinsic Reward RL Algorithm Based on Parallel Architecture. Because the behavior strategy is inconsistent with the target strategy, which makes the algorithm become an off-policy algorithm. Therefore, the collected data needs to be processed before learning; otherwise, the training process will generate a large deviation resulting in performance degradation. In order to solve this problem, we use the V-trace method to process the sampled intrinsic rewards and extrinsic rewards and obtain the estimated value function that the learner can utilize to update policy.

According to behavior policy μ, a worker interacts with the environment and collects an episode of data. We define the n-step V-trace value of state x_s as follows:

$$v_s = V(x_s) + \sum_{t=s}^{s+n-1} \gamma^{t-s} \left(\prod_{i=s}^{t-1} c_i \right) \sigma_t V, \tag{9}$$

where $\sigma_t V = \rho(r_t + \gamma V(x_{t+1}) - V(x_t))$ is the temporal difference value of V estimated from data sampled by the worker.

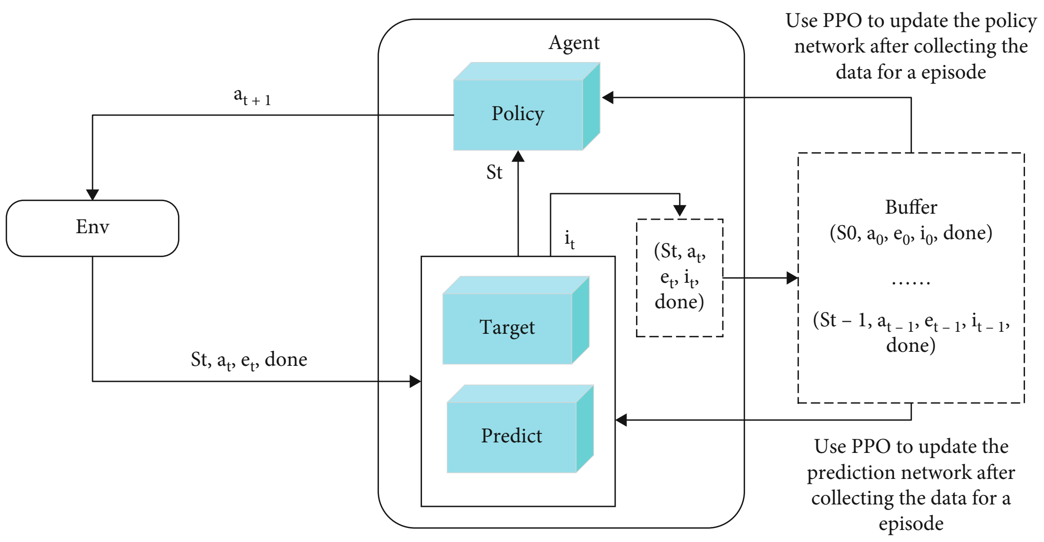
Agent
Policy
a_{t+1}
St
Env
i_t
(S_t, a_t, e_t, i_t, done)
Target
Predict
St, a_t, e_t, done
Use PPO to update the policy network after collecting the data for a episode
Buffer
(S0, a_0, e_0, i_0, done)
……
(St − 1, a_{t-1}, e_{t-1}, i_{t-1}, done)
Use PPO to update the prediction network after collecting the data for a episode

FIGURE 4: Architecture diagram of the reinforcement learning algorithm based on intrinsic reward.

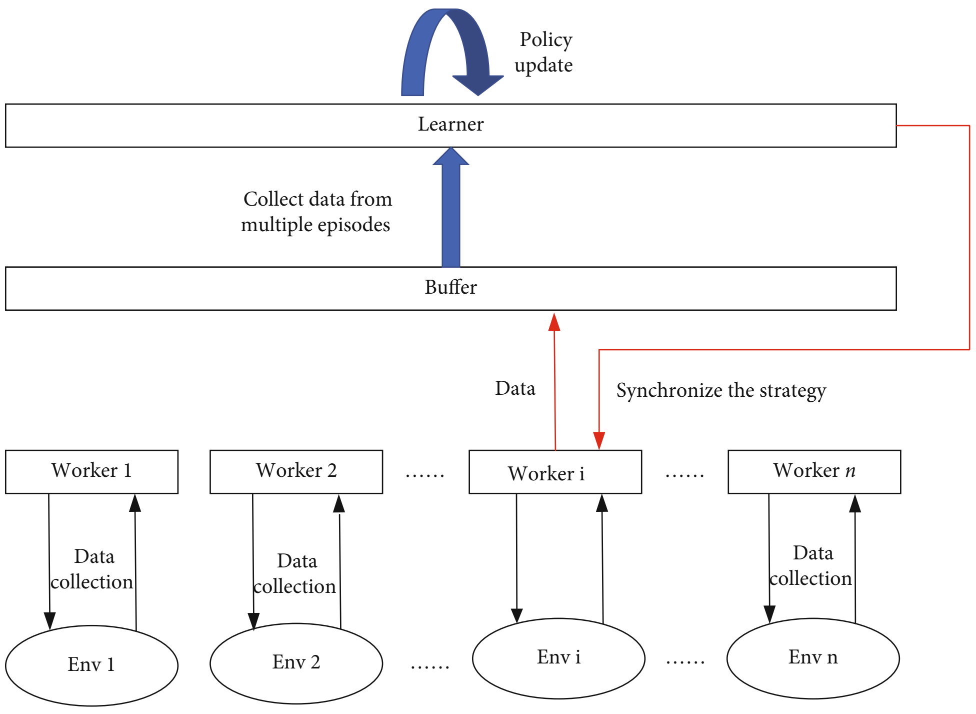

FIGURE 5: Architecture diagram of Impala.

Parameter $\rho_t = \min\left(\rho, \left(\pi(a_t|x_t)/\mu(a_t|x_t)\right)\right)$ and parameter $c_i = \min\left(c, \left(\pi(a_t|x_t)/\mu(a_t|x_t)\right)\right)$. ρ_t and c_i have different roles in the equation. The former is used in the definition of $\sigma_t V$ and according to the fixed point theory, the estimated value function of the above formula is the value function V^{π_ρ} under the strategy π_ρ. The policy π_ρ can be written as follows:

$$\pi_\rho(a|x) = \frac{\min\left(\rho\mu(a|x), \pi(a|x)\right)}{\sum_{b\in A} \min\left(\rho\mu(b|x), \pi(b|x)\right)}. \qquad (10)$$

The above formula shows that when ρ is infinite, policy π_ρ is the target policy and V^{π_ρ} is the value function of the target policy π. If ρ is close to zero, we obtain the value function of the behavior policy V^{π_μ}. When we choose a truncation level $\rho < \infty$, our fixed point is the value function V^{π_ρ} which is somewhere between μ and π. The weights $c_s \cdots c_{t-1}$ quantitatively assess the impact of temporal difference $\sigma_t V$ observed at time t on the update of the value function at a previous time s. Thus, in the formula, ρ impacts the convergence fixed point of the value function and c impacts the coverage speed.

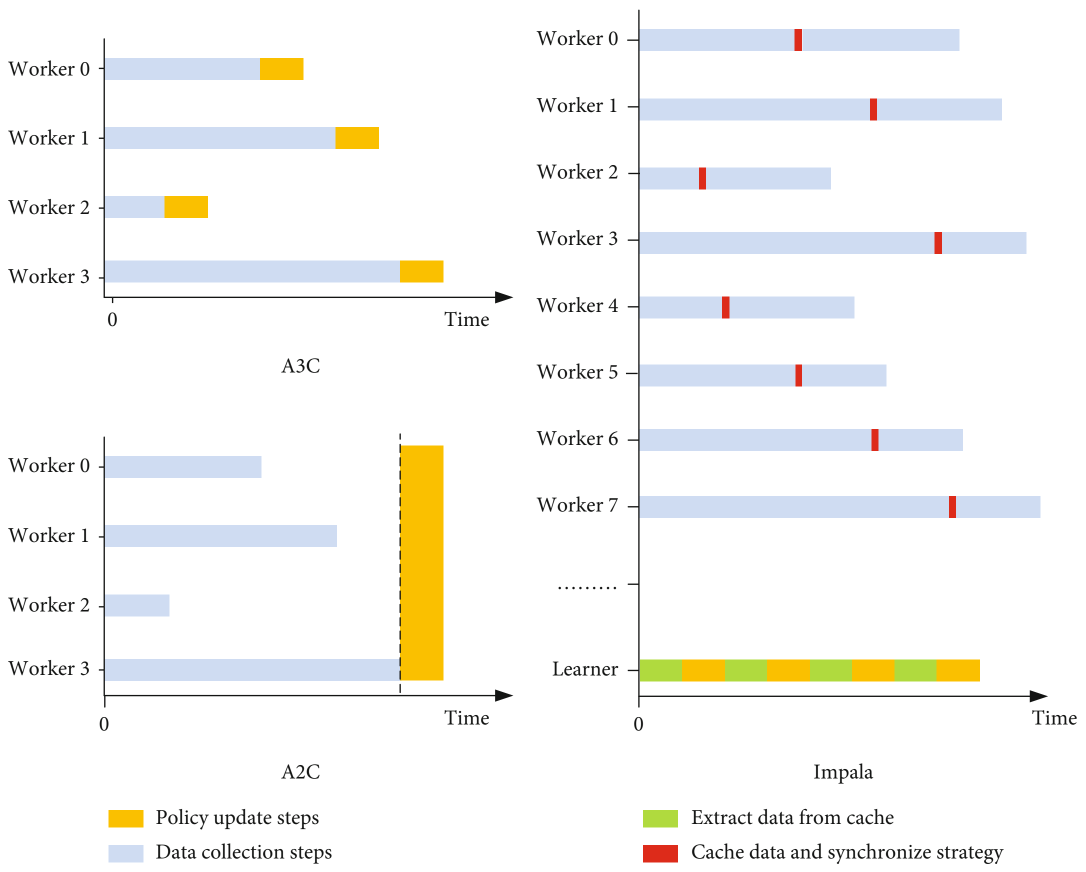

FIGURE 6: Timeline comparison chart.

Finally, we propose the intrinsic reward RL algorithm under the parallel framework and the flow chart is shown in Figure 7. It is shown that each worker independently interacts with its environment to collect data and store it in the buffer. Then, the worker synchronizes the current latest policy from the learner and starts the next round of interaction. The learner will periodically fetch data from the buffer for policy updates. The data collected by different behavior strategies will be processed through the V-trace method. Then, the value function of the intrinsic reward and the value function of the extrinsic reward will be calculated. Finally, the policy network and the value function evaluation network are updated by using the PPO algorithm. Our proposed algorithm can be divided into two parts, worker and learner. The worker is mainly responsible for the data acquisition function, and its pseudocode can be found in Algorithm 1. Learner is mainly used for policy training, and its pseudocode can be used in Algorithm 2.

4. Experiments

In this section, we will set the parameters used in the experiments and define the agent's action space, state space, and reward function. We will then analyze the experimental results.

4.1. Parameter Settings. We use OpenAI Gym [18] to build the simulation environment. We assume a communication assurance task, and the task is set as follows. In a three-dimensional map with the size of $50\,\text{km} \times 50\,\text{km} \times 5\,\text{km}$, there are randomly distributed buildings with the height of $50\,\text{m} \sim 150\,\text{m}$. The NTN platform provides wireless coverage for ten moving IoT devices and ten static IoT devices randomly distributed on the ground. We set all IoT devices to perform burst traffic with a probability of 0.01 every time slot. It can also be regarded as a Poisson distribution with a λ of 1 in 100 seconds. The NTN platform can adjust the flight direction within the range of 0–360 Â°, the flight elevation angle within the range of 0–180 Â°, and the flying speed within the range of $180 \sim 300\,\text{km/h}$. Then, we will define the state space, action space, and reward function in the reinforcement learning process.

4.1.1. The State Space. The agent's state space is the combination of the NTN platform and 20 IoT devices' states. The reason why we only set up 20 devices in the experiment is because it is limited by the performance of individual experimental devices. More users means that UAV-APs need to cover more locations and actively adjust their flight status to provide users with high-quality services. In this experiment, since the starting positions and flight trajectories of each user are random, the positions are scattered and the trajectories are uncertain, which ensures the validity of the experimental results to a certain extent. In the future, we will add more users to test the performance of the algorithm. The NTN platform state can be defined as a 7-

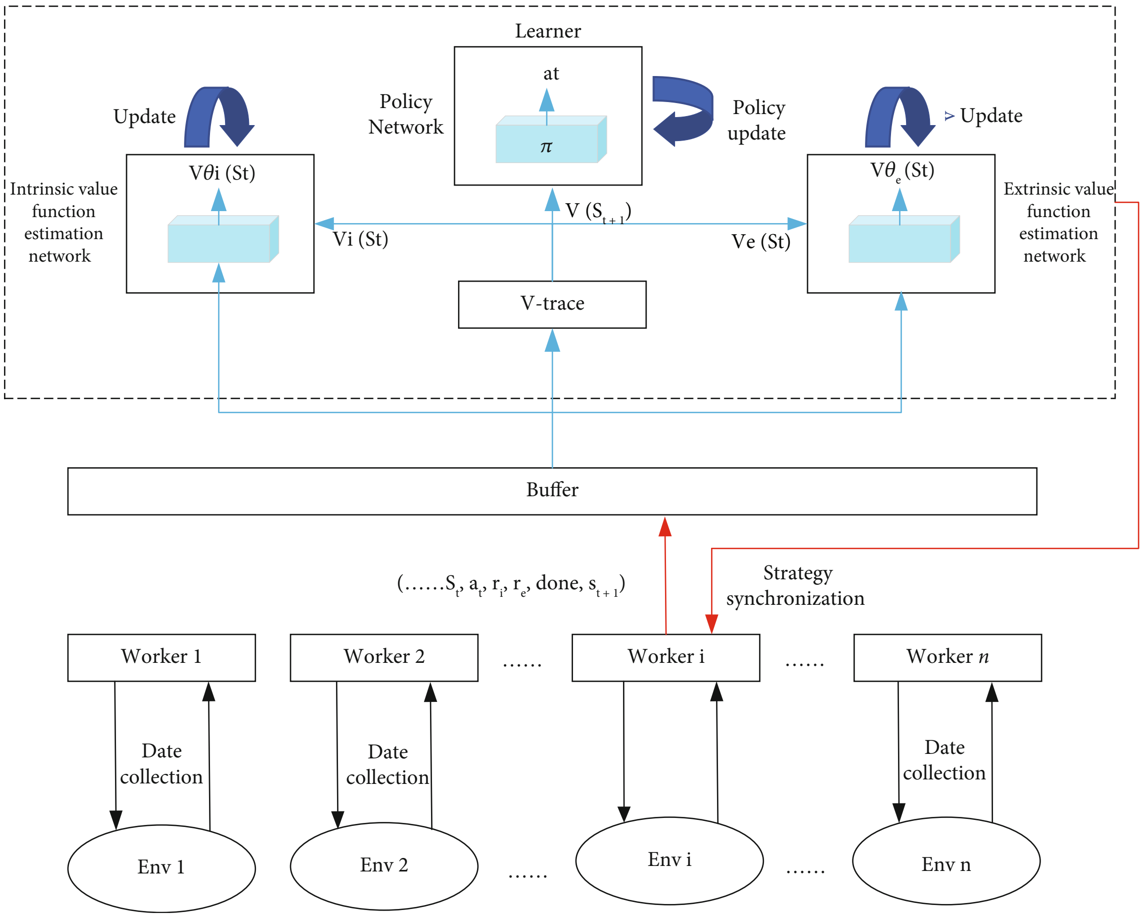

Figure 7: Algorithm flow chart.

element vector that includes the NTN platform's location coordinates in the map, the flight angle, flight elevation angle, speed of NTN platform, and base station transmit power. The IoT device state is a 4-dimensional vector including the device's coordinates on the map, moving angle, and speed of IoT device. In order to ensure the consistency of the matrix form, the remaining elements are set to 0. So, the state space of the agent is the 21×7 matrix.

4.1.2. The Action Space. The action space of the agent is a vector containing four continuous elements, including speed, flight angle, flight elevation angle of the NTN platform, and base station transmit power. We need to standardize these variables, and then, the agent will execute these actions in a simulated environment and get the reward at this time slot.

4.1.3. The Reward. The NTN platform needs to ensure that the downlink rate of each IoT device is greater than the threshold rate while preventing fast fading caused by the Doppler frequency shift. Under this premise, the goal of the task is to maximize the average downlink rate of the device under adequate communication time. We set the reward value for each time slot as the average downlink rates of IoT devices performing bursty services in that time slot. However, if a device is affected by fast fading or the downlink rate is lower than a threshold, the downlink speed of that device is considered to be is 0. Besides, if the NTN platform hits the building, we terminate the experiment and return -100 as the reward value for this round. If the problems mentioned above do not occur during the communication guarantee task, the reward value of 100 is returned. If the experiment termination condition has not been triggered within time t, we terminate the experiment and return the reward value 100.

The algorithm code is written in Python 3.8, mainly using libraries such as PyTorch, Gym, and NumPy. The training environment is the simulation environment introduced in Section 3, built on the Windows 10 system with 2 NVIDIA 3090 graphic cards and 64 g RAM. In the experiments, we tested the proposed algorithm with the current mainstream reinforcement learning algorithms and algorithms under different parallel architectures in a simulation environment. Each experiment ran on 2 K rollouts with 32 parallel environments, and the hyperparameters used in the algorithm are shown in Table 1. The parameters of the communication metrics have been introduced in Section 2 and the introduction of algorithm hyperparameters in Section 3. We utilize 32 parallel environments for data collection to improve the training speed. At the same time, we

```
$T$←numbers of parallel environments; $K$←→initial length of rollout;
$D$←number of initial steps for initializing observation;
t=0
Sample state $s_0 \sim P_0(s_0)$
For d=1 to D do
  Sample $a_t \sim Normalized(a_t)$
  Sample $s_{t+1} \sim p(s_{t+1}|s_t, a_t)$
  Update observation normalization parameters using $s_{t+1}$
  t += 1
**End for**
**While true**
t=0
  For j=1 to K do
    For h=1 to T do
      Sample $a_t \sim Normalized\,\pi(a_t|s_t)$
      Sample $s_{t+1}, e_t \sim p(s_{t+1}, e_t|s_t, a_t)$
      Caculate intrinsic reward $i_t = \|f^\wedge(s_{t+1}) - f(s_{t+1})\|^2$
      Add $s_t, s_{t+1}, a_t, e_t, i_t$ to optimization batch $B_h$
      Update reward normalization parameters using $i_t$
      t +=1
    **End for**
    Collect data,put $B_h$ into the global buffer $B$
  **End for**
  Synchronize the strategy from the learner
```

Algorithm 1:Data acquisition module pseudocode.

```
L←the length of the data extracted; $N_{opt}$←→initial number of optimization steps;
$K$←initial length of rollout; $N_{opt}^\wedge$←→number of optimization steps in later stages;
$K^\wedge$ ←→new length of rollout in later stages;
**While true**
  Extract data from the buffer
  Normalize the intrinsic rewards contained in B
  For j=1 to L do
    Caculate returns $R_{I,i}$ and advantages $A_{I,i}$ for intrinsic reward
    Caculate returns $R_{E,i}$ and advantages $A_{E,i}$ for extrinsic reward
    Utilize $A_{I,i}$ and $A_{E,i}$ by using V-trace and obtain value function $A_{I,i}^\wedge$ and $A_{E,i}^\wedge$ of $\pi_\rho$
    Caculate combined advantages $A_i = A_{I,i}^\wedge + A_{E,i}^\wedge$
    Update observation normalization parameters
  **End for**
  **For** j=1 to $N_{opt}$ do
      Optimize $\theta_\pi$ wrt PPO loss on batch $B, A_i$ using AdamW
      Optimize $\theta_f$ wrt distillation loss on $B$ using AdamW
  **End for**
  Check recent data if meet the conditions of adjusting parameters
    Then set $K = K^\wedge, N_{opt} = N_{opt}^\wedge$
```

Algorithm 2: Policy training module pseudo code.

set the initial rollout length to 2,500 and the initial optimization step to 4 to pretrain the parameters of the model. We set the learning rate to 0.0003 to avoid overfitting and set the discount values for internal and external rewards to 0.99 and 0.999 to ensure that the agent is not affected by reward values that are too old. The design of GAE and PPO parameters is to avoid that the optimization step size is too large, resulting in failure to converge to a better result.

Figure 8 shows the number of IoT devices that can operate normally after adopting the NB-IoT method or the method based on the NTN communication platform. Since an IoT device obeys a Poisson distribution with a λ of 1 within 100 seconds, the data are counted in units of 100 seconds and all are rounded.

The blue bar in represents the number of mobile IoT devices that successfully performed burst services within

Table 1: The value of parameter used in the algorithm.

Hyperparameter	Value
Number of users (M)	10
Threshold rate (C_{th})	5 M/s
Bandwidth (W)	10 MHz
Transmit power (P_s)	200 W
Path loss parameter ($\text{PL}_{\text{LoS}}/\text{PL}_{\text{NLOS}}$)	0.1/21.0
Environment parameters (a, b)	0.1750
Parallel environments (T)	32
Number of rollouts (N)	2000
Initial rollout length (K)	2500
Initial optimization steps (N_{opt})	4
Initial learning rate	0.0003
Initial number of minibatches	8
Intrinsic discount factor	0.99
Extrinsic discount factor	0.999
Optimization algorithm	AdamW
GAE factor λ	0.95
PPO clip range	[0.9,1.1]
Coefficient of extrinsic reward	1
Coefficient of intrinsic reward	1
ρ	1
c	1

100 s. The orange bar represents the number of static IoT devices that successfully performed burst services within 100 s. According to the probability calculation, both static IoT devices and dynamic IoT devices will perform an average of 10 burst services within 100 s. It shows the results using the NB-IoT method. It can be seen that the static IoT device and the mobile device can successfully execute the burst service six times and three times, respectively, every 100 seconds, which is far less than the estimated execution times. The reason is that NB-IoT relies on fixed base stations, the signal strength will attenuate as the distance increases, and buildings will also block the signal. Attenuation of the signal makes the downlink rate of static and mobile IoT devices far away from the base station too low to perform services typically. At the same time, the fixed base station cannot effectively handle the fast fading caused by the Doppler effect, so the support for mobile IoT devices is even worse.

The lower diagram in Figure 8 represents the use of the NTN communication platform to provide wireless coverage for IoT devices. It can be seen that the static IoT device and the mobile device can successfully execute 10 and 8 burst services, respectively, every 100 seconds, which is relatively close to the estimated execution times. This is because the NTN platform can adjust its flight status according to the learned strategy based on the current status information and IoT devices. Learned strategies enable the NTN platform to take into account both static devices and mobile devices and maximize the average downlink rate of all IoT devices on the premise of avoiding fast fading. On the other hand, in the experiment, the method based on the NTN communication platform can make the average downlink speed of IoT devices reach 10 Mbps. In contrast, the NB-IoT method can only provide a maximum downlink rate of 250 Kbps. Therefore, the method based on the NTN platform can provide services for more IoT devices and significantly increase the downlink rate of the device.

Figure 9 shows the performance comparison between our algorithm and current mainstream reinforcement learning algorithms in the NTN platform path planning task. The results show that the algorithm performs poorly in the early stage. This is because the agent needs to fully explore the environment under the guidance of internal rewards in the early stage and accumulate more valuable experience data. Therefore, in the early stage of training, the agent tends to explore rather than improve the policy performance. We can see that after training to nearly 700 scenes of data, the algorithm's performance will increase exceptionally quickly. This is because the agent has conducted a comprehensive exploration of the environment and has begun to use the collected experience data to make targeted improvements. At this stage, the agent is more inclined to improve the policy performance. It can be seen that compared with the current mainstream algorithms, our algorithm has a faster convergence speed and a higher final score.

On the other hand, the learning rate of the proposed algorithm is much faster than other algorithms. When the agent is trained to 1,000 episodes, the algorithm's performance decreases. This may be due to overfitting caused by the large update step size. Subsequently, the algorithm finally completed the convergence at 1,200 episodes with the policy iteration. It can be seen that our algorithm has a higher final score than several other mainstream algorithms. This is because the agent collects diverse empirical data in the early stage, preventing the algorithm from converging to a locally optimal solution. On the other hand, our proposed algorithm also has the fastest convergence speed. This benefits from the improved efficiency of agent-environment interaction brought by the decoupled parallel architecture.

Figure 10 shows the performance comparison results of the proposed algorithm. It can be seen that the proposed algorithm has better performance than the algorithms based on A2C and A3C architecture. Among them, the performance of the A3C algorithm is the worst. Due to the asynchronous update method, each worker uses a different strategy leading to accumulation of deviations in the update process. This makes the algorithm based on A3C have poor performance and significant fluctuation during the training process. Compared to the algorithm based on the A2C architecture, the final performance is close but our algorithm allows the algorithm to reach the target score faster. The reason is that the worker and the learner are wholly decoupled, which makes the worker not need to wait for the learner to update the strategy improving the sampling efficiency. Comparing the algorithm without V-trace correction, it can be found that the algorithm without V-trace correction has significant fluctuation and hardly learns effect policy. The target policy will have a significant deviation if the learner uses data that V-trace has not corrected to update the

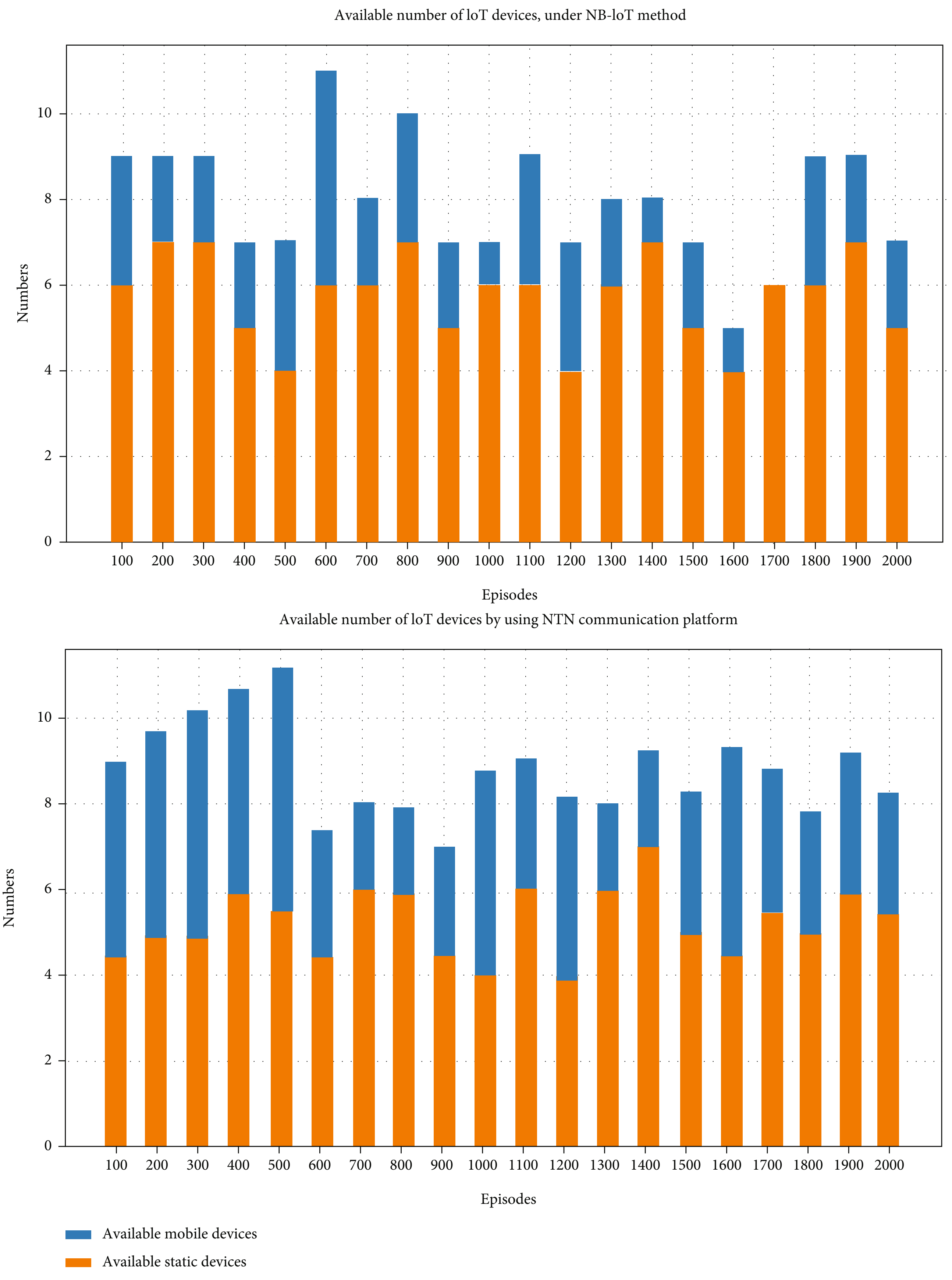

Figure 8: Comparison of the number of available IoT devices.

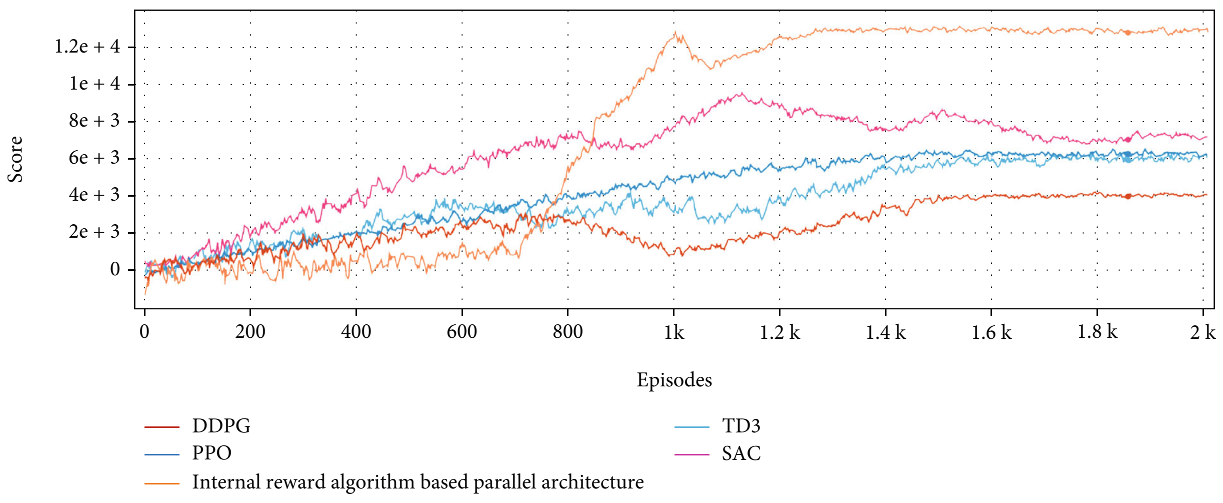

Figure 9: Performance comparison between the proposed algorithm and mainstream reinforcement learning algorithms.

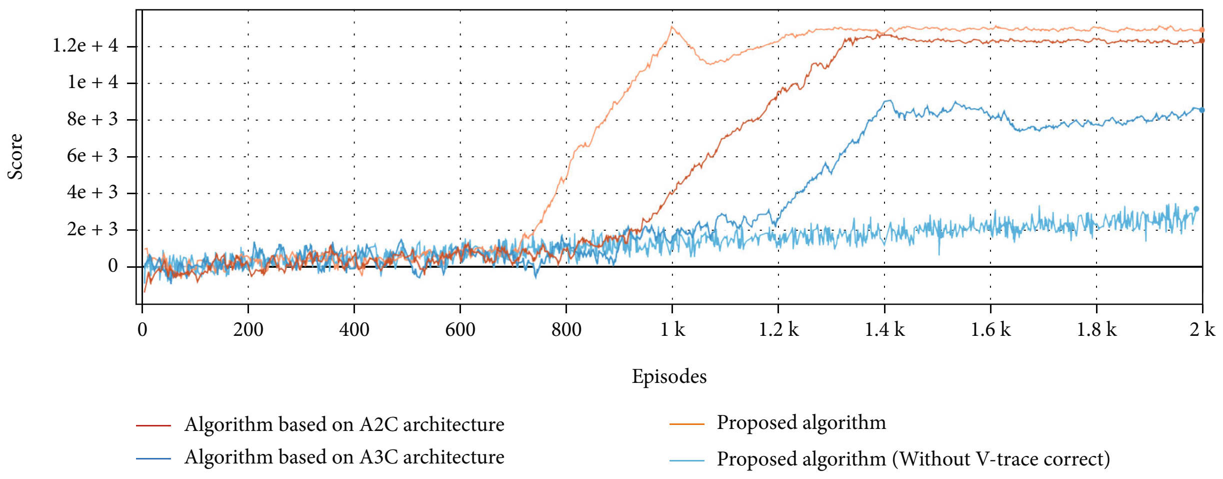

Figure 10: Performance comparison between the proposed algorithm and algorithms under other parallel architectures.

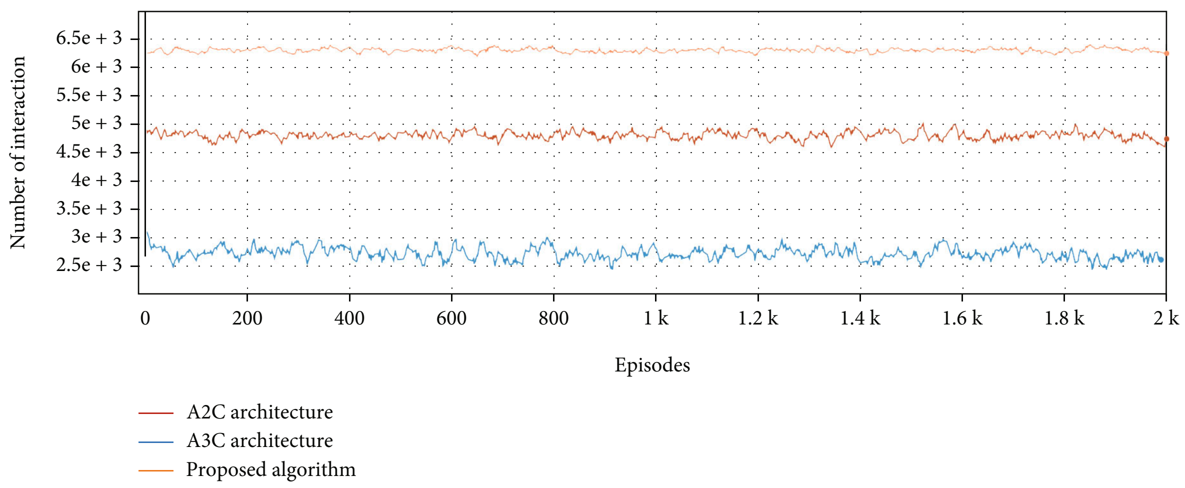

Figure 11: Comparison of the number of interactions between the agent and the environment per second in one episode.

strategy. The accumulation of deviation will reduce the accuracy of the agent's prediction of the value function, eventually leading to poor performance. So, the experiment proves that the V-trace correction is significant for the off-policy algorithm.

Figure 11 shows the average number of interactions between the agent and the environment per second in an episode of data during the training process. Our algorithm has the most significant number of interactions per second and has a minor variance during the training process, which means that the interaction is stable. The difference in the interaction time of different algorithms mainly depends on when the worker waits for the learner to update the strategy and synchronize the strategy. In A3C, each worker needs to wait for the learner to update its policy using collected data. Multiple workers queue up so that congestion occurs and the speed of interaction decreases. In A2C architecture, the learner needs to wait for all workers to complete the interaction and store the data in the buffer before starting the policy update. The main reasons for the weak interaction are the time that the learner waited for the workers and the time that the worker waited for the learner to process large-scale data. In our algorithm, the learner and workers are entirely decoupled and work asynchronously. Therefore, the learning speed of the learner does not affect the interaction between the workers and the environment. This is why our algorithm has the most significant number of interactions per second and the most stable interaction process.

5. Conclusion

This paper mainly explores the possibility of using a nonterrestrial IoT network built with the NTN platform to provide high-quality wireless coverage for terrestrial mobile IoT devices. The real-time path planning of the NTN platform is realized by using the reinforcement learning algorithm, which effectively improves the downlink rate of ground mobile IoT devices. Experimental results showed that our algorithm significantly improved the final performance and learning rate compared to mainstream reinforcement learning algorithms. Our architecture has higher data collection efficiency than commonly used parallel architectures. In summary, the NTN IoT network has the potential to make up for the shortage of fixed base stations that cannot provide high-quality signal coverage for ground IoT devices in border areas and urban hotspots. At the same time, this will further promote the development and application of IoT. In order for the algorithm to be applied in a practical environment, we need to solve the following problems in the future. First, the impact of signal interference on the NTN platform needs to be resolved. Secondly, the NTN platform needs to find a method that can quickly and accurately locate moving targets. According to the research, Liu et al. use federated learning to train a distributed network to identify signal disturbances [19]. Simulation results show that the method leads to excellent recognition performance with a small dataset. On the problem of moving target localization, Liu et al. propose a passive position parameter estimator for estimating moving aerial targets using multiple satellites [20] and an intelligent passive detection method based on reservoir computing networks [21]. The simulation results show that the method can accurately estimate the position parameters of moving objects and achieve efficient detection of moving objects. We will refer to the content of the above articles in the future and improve the algorithm proposed in this article to adapt to practical application scenarios.

Data Availability

This article uses the OpenAI Gym framework to build a reinforcement learning simulation environment. All experimental analysis data were obtained from this platform. The simulation platform has been applied for the corresponding software copyright, numbered 2021SR1463151. Since the research results are supported by the Academy of Military Science of the PLA, all experimental data and results are kept by the institution. You can contact Mr. Zhang at lanyangyang_1994@sina.com to apply for the experimental data.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This study is supported by the National Natural Science Foundation of China (61971092).

References

[1] M. Chen, Y. Miao, Y. Hao, and K. Hwang, "Narrow band internet of things," *IEEE Access*, vol. 5, pp. 20557–20577, 2017.

[2] H. Han, L. Fang, W. Lu, W. Zhai, Y. Li, and J. Zhao, "A gcica grant-free random access scheme for m2m communications in crowded massive mimo systems," *IEEE Internet of Things Journal*, vol. 9, no. 8, pp. 6032–6046, 2022.

[3] "TR 38.821-V0.4.0. "Solutions for NR to support non-terrestrial networks (NTN)"," https://www.3gpp.org/DynaReport/38821.html, 2019.

[4] J. Guo, Y. Huo, X. Shi et al., "3D aerial vehicle base station (UAV-BS) position planning based on deep Q-learning for capacity enhancement of users with different QoS," *Requirements[C]// 2019 15th International Wireless Communications and Mobile Computing Conference (IWCMC)*, p. 23, 2019.

[5] H. Bayerlein, P. D. Kerret, and D. Gesbert, "Trajectory Optimization for Autonomous Flying Base Station via Reinforcement Learning[C]," in *2018 IEEE 19th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, p. 24, IEEE, 2018.

[6] V. Mnih, K. Kavukcuoglu, D. Silver et al., "Playing Atari with deep reinforcement learning," *Computer Science*, vol. 25, 2013.

[7] Q. Wang, W. Zhang, Y. Liu, and Y. Liu, "Multi-UAV dynamic wireless networking with deep reinforcement learning," *IEEE Communications Letters*, vol. (99), pp. 1–22, 2019.

[8] H. V. Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double Q-learning," *Computer Ence*, vol. 26, 2015.

[9] C. H. Liu, X. Ma, X. Gao, and J. Tang, "Distributed energy-efficient multi-UAV navigation for long-term communication coverage by deep reinforcement learning," *IEEE Transactions on Mobile Computing*, vol. 19, no. 6, pp. 1274–1285, 2020.

[10] H. Qi, Z. Hu, H. Huang, X. Wen, and Z. Lu, "Energy efficient 3-D UAV control for persistent communication service and fairness: a deep reinforcement learning approach," *IEEE Access*, vol. 8, no. 53172-53184, pp. 53172–53184, 2020.

[11] T. P. Lillicrap, J. J. Hunt, A. Pritzel et al., "Continuous control with deep reinforcement learning," *Computer Ence*, vol. 27, 2015.

[12] ITU-R, *Rec. P. 1410-2 Propagation Data and Prediction Methods for the Design of Terrestrial Broadband Millimetric Aadio Access Systems, P Series*, Radiowave propagation, 2003.

[13] A. Saakian, *Radio Wave Propagation Fundamentals, LI Ji-Jun; CHEN Hai-yan*, Propagation properties of metal clad waveguide at communication frequency, 2020.

[14] R. S. Sutton and A. G. Barto, "Reinforcement learning," *A Bradford Book*, vol. 15, no. 7, pp. 665–685, 1998.

[15] Y. Li, "Deep reinforcement learning: an overview," https://arxiv.org/abs/1701.07274, 2017.

[16] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, *Proximal Policy Optimization Algorithms*, 2017.

[17] Y. Burda, H. Edwards, A. Storkey, and O. Klimov, *Exploration by Random Network Distillation*, 2018.

[18] G. Brockman, V. Cheung, L. Pettersson et al., *OpenAI Gym*, 2016.

[19] M. Liu, Z. Liu, W. Lu, Y. Chen, X. Gao, and N. Zhao, "Distributed few-shot learning for intelligent recognition of communication jamming," *IEEE Journal of Selected Topics in Signal Processing*, vol. 16, no. 3, pp. 395–405, 2022.

[20] M. Liu, B. Li, Y. Chen et al., "Location parameter estimation of moving aerial target in space–air–ground-integrated networks-based iov," *IEEE Internet of Things Journal*, vol. 9, no. 8, pp. 5696–5707, 2022.

[21] M. Liu, C. Liu, M. Li, Y. Chen, S. Zheng, and N. Zhao, "Intelligent passive detection of aerial target in space-air-ground integrated networks," *China Communications*, vol. 19, no. 1, pp. 52–63, 2022.