Joint Beam-Hopping Scheduling and Coverage Control in Multibeam Satellite Systems

Guoliang Xu, Feng Tan, Yongyi Ran, Yanyun Zhao,and Jiangtao Luo

***Abstract*—Beam-hopping scheduling technology is a crucial technology for improving the performance of resource-constrained multibeam satellite (MBS) systems. However, it is challenging to dynamically match the communication demands of the terrestrial cells with the beam transmission capacity. Almost all state-of-art beam-hopping scheduling approaches only consider a fixed beam coverage radius, which always results in a waste of transmission resources. In this letter, we propose a deep reinforcement learning-based algorithm to jointly optimize the beam-hopping scheduling and coverage control (called DeepBeam) which flexibly uses three degrees of freedom of time, space and beam coverage radius. In addition, to reduce the computational complexity and accelerate the training of DeepBeam, we first train a single model for one of the beams and then use this trained model for each beam in a polling manner to determine the cell to be illuminated and the beam coverage. Furthermore, the simulation results illustrate that the proposed DeepBeam can improve not only the throughput of MBS but also reduce the packet loss probability.**

***Index Terms*—Multibeam satellite, Deep reinforcement learning, Beam-hopping scheduling, Coverage control.**

I. Introduction

DUE to the scarcity and high cost of satellite communication resources, improving the resource utilization is essential to meet the dramatic increase in communication demands [1]. Beam-hopping (BH) technology is promising to provide flexibility in resource management for multibeam satellite systems (MBS) [1]. However, the current BH schemes only determine which cell needs to be illuminated at each time-slot without considering the coverage radius of the illuminated beams [2]. Since the communication demands of each cell are time-varying, a fixed beam coverage radius may cause a waste of satellite transmission resources under a low load or cause congestion under a high load [3]. Therefore, it is critical to not only regulate beam-hopping but also control its coverage radius.

In order to effectively match the beam resources with non-uniform communication demands, researchers have developed many beam-hopping scheduling strategies. Some researchers have adopted offline algorithms [4–6] to design beam hopping strategies. Specifically, the beam-hopping strategy based on genetic algorithm [4], cuckoo algorithm [5], and greedy algorithm [6] are adopted to improve the performance of MBS. But as the number of beams increases, the computational complexity of the above dynamic optimization method becomes unmanageable [7]. To deal with the above problems, many researchers utilized deep reinforcement learning (DRL), an online algorithm, to design beam-hopping scheduling strategies that have been shown to perform well [7–9]. However, these works mainly focused on finding the best beam-hopping scheduling strategy without taking full advantage of the flexibility of beam coverage radius. Zhang *et al.* in [10] proposed a beam resource management algorithm to improve system performance by adjusting beam coverage but does not consider the beam-hopping scheduling in multibeam satellites, which limits the obtained conclusions.

In this letter, we propose a joint optimization algorithm of beam-hopping scheduling and coverage control (BHCC) based on DRL, which makes full use of the flexibility of beam coverage radius(called DeepBeam). In DeepBeam, we first formulate an objective optimization problem to maximize the system throughput. Then, the target problem is modelled as a Markov decision process (MDP) considering the stochastic communication demands. In addition, to reduce the computational complexity and accelerate the training of DeepBeam, we first train a single model for one of the beams and then use this trained model for each beam in a polling manner to determine the cell to be illuminated and the beam coverage radius. Finally, it can be seen from the simulation results that the DeepBeam algorithm proposed in this letter has better performance than other algorithms. In particular, compared with the DRL-based algorithm without consider the adaptive beam coverage, Deepbeam has a smaller packet loss probability due to its adaptive beam coverage ability.

II. System Model and Problem Formulation

This paper considers a GEO satellite working in the Ka-band. Since the GEO satellite is stationary relative to the ground, the satellite does not consider its mobility of the satellite. Assume it carries K beam transmitters. We divide the entire satellite coverage area into N equal-sized cells and mark them according to the latitude and longitude coordinates, assuming that there is a satellite ground station in the center of each cell to receive and transmit communication demand. In addition, each satellite has a buffer to record the communication demands of each cell at different slots. In the MBS, the number of beams is much smaller than the number of cells ($K \ll N$) and a beam can cover and serve a cell cluster (a

This work was supported by the National Science Foundation of China (No.62003067, 62101525, 62171072, 62172064), and Natural Science Foundation of Chongqing (cstc2021jcyj-msxmX0586). (*Corresponding author: Yongyi Ran and Jiangtao Luo.*)

Guoliang Xu, Feng Tan, Yongyi Ran, Yanyun Zhao, and Jiangtao luo are with the School of Communication and Information Engineering, Chongqing University of Posts and Telecommunications, Chongqing, 404100 China (e-mail: xugl@cqupt.edu.cn, 2016215052@stu.cqupt.edu.cn, ranyy@cqupt.edu.cn, S200101135@stu.cqupt.edu.cn, luojt@cqupt.edu.cn).

cell cluster can consist of one or more cells). We denote the set of beams as $\kappa = \{k | k = 1, 2, ..., K\}$ and the set of cells as $C = \{c_n | n = 1, 2, ..., N\}$. The number of data packets to be served recorded in the satellite buffer of cell c_n at t is $\Upsilon_t^{c_n} = \{\phi_{t,l}^{c_n} | l = 0, 1, ..., l_{th+1}\}$, where l is the queuing delay of the data packet, and l_{th} is the maximum tolerable queuing delay of the data packet. The number of data packets requested by cell c_n at t is $\phi_{t,0}^{c_n}$, which satisfies Poisson distribution. The packet will be discarded if the queuing delay exceeds the maximum tolerable queuing delay, i.e., $l = l_{th+1}$.

The number of unserved packets $\lambda_t^{c_n}$ in c_n at t is correlated between adjacent moments as follows

$$\lambda_t^{c_n} = \lambda_{t-1}^{c_n} + \phi_{t,0}^{c_n} - \wp_t^{c_n} - \phi_{t,l_{th+1}}^{c_n}, \tag{1}$$

where $\lambda_{t-1}^{c_n}$ the number of unserved packets in cell c_n at $t-1$, i.e., $\lambda_{t-1}^{c_n} = \sum_{l=0}^{l=l_{th}} \phi_{t-1,l}^{c_n}$. The $\wp_t^{c_n}$ means the number of data packets transmitted at t.

According to ITU-R S.672-4 [11], the relationship between angle θ and transmit antenna gain is denoted as

$$G_{tx}(\theta) = \begin{cases} G_m, & \theta < \theta_b \\ G_m - 3(\theta/\theta_b)^2, & \theta_b \le \theta \le a\theta_b \\ G_m + L_s, & a\theta_b \le \theta \le b\theta_b \\ \max\left\{G_m + L_s + 20 - 25\log_{10}\left(\frac{\theta}{\theta_b}\right), 0\right\}, & \text{else} \end{cases} \tag{2}$$

where $a = 2.88$, $b = 6.32$, $L_s = -25$ dB, θ denotes the off-axis angle in degrees and θ_b denotes the 3 dB beamwidth. G_m denotes the maximum gain of satellite transmit antenna [9], which has a form as

$$G_m = 10\log_{10}\left[4.93\left(\frac{70}{\theta_b}\right)^2\right]. \tag{3}$$

It can be seen from Eq.(3), that increasing the half-lobe gain angle of the beam will reduce the maximum gain of the transmitting antenna, and the beam half-lobe gain angle determines the beam coverage, so the beam coverage and beam capacity are negatively correlated.

We approximate the beam capacity from beam k to the cell cluster C_k it covers as the beam capacity from beam k to the center cell c_n of the cell cluster. The channel gain from beam k to cell c_n can be expressed as $h^{k,n} = G_{tx}G_rL_f$, where G_r is the receive antenna gain and L_f is the free space loss. Furthermore, When a beam serves a cell on the ground, the beam adopts a full-frequency multiplexing scheme, namely reuse factor = 1. So, the channel capacity from beam k to cell c_n can be expressed as

$$F_k^{c_n} = \varepsilon_k^{c_n} W_k \log_2\left(1 + \frac{h^{k,n} \cdot P_k}{W_k N_0 + \sum_{i \in \kappa, i \neq k} h^{i,n} \cdot P_i}\right), \tag{4}$$

where P_k and P_i is the transmit power of the beam k and beam i , N_0 represents white Gaussian noise, W_k represents the bandwidth of the beam k, $\varepsilon_k^{c_n}$ is used to judge whether the cell c_n is the center of the cell cluster C_k covered by the beam k, if so, $\varepsilon_k^{c_n} = 1$, otherwise, $\varepsilon_k^{c_n} = 0$.$\wp_t^k$ is the number of data packets transmitted by the beam k for the cell cluster C_k whose center cell is c_n at t, which can be expressed as

$$\wp_t^k = \min\{F_k^{c_n}, \lambda_t^k\}, \lambda_t^k = \sum_{\text{n}=1}^{N} \lambda_t^{c_n}, if \quad \varepsilon_k^{c_n} = 1, \tag{5}$$

where λ_t^k is the sum of packets to be transmitted in all cells in the cell cluster C_k covered by the beam k.

The $\wp_t^{c_n}$ can be expressed as

$$\wp_t^{c_n} = \wp_t^k / \text{Z}, \forall c_n \in C_k \tag{6}$$

where Z is the number of cells contained in the cell cluster C_k.

The purpose of BHCC is to dynamically select beam coverage strategies so that more communication demands are transmitted rather than cleared by satellite buffers to maximize system throughput. Therefore, the optimization problem is mathematically formulated as

$$\begin{aligned} \max \quad & P1 = \sum_{n=1}^{N} \wp_t^{c_n}, \\ s.t. \quad & \text{C1} : \sum_{k=1}^{K} P_k \le P_{tot}, \\ & \text{C2} : P_k \le P_{\max}, \\ & \text{C3} : \chi_t^k \le R_{\max}, \forall k, t, \end{aligned} \tag{7}$$

where $P1$ is to maximize system throughput. The constraint $C1$ indicates that the sum of the power allocated to all beams should not exceed the total system power. Constraint $C2$ means that the transmit power of a single beam must not exceed the maximum power that a single beam can carry. $C3$ requires each beam size to be less than a threshold and χ_t^k denotes the coverage radius of the beam k at t.

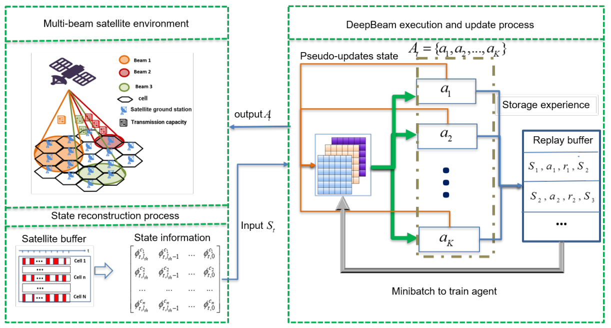

Fig. 1. The framework of DeepBeam for multi-beam satellite system.

III. Joint Optimization Algorithm of Beam-Hopping Scheduling and Coverage Control

In this section, the joint optimization algorithm of beam-hopping scheduling and coverage control based on deep reinforcement learning is introduced in detail. Firstly, we describe the objective optimization problem as an MDP and build a Markov model. Then, we introduce the the architecture of DeepBeam algorithm.

A. MDP Model

According to Eq.(1), the communication demands of each cell are related at adjacent moments, and the state of the next moment is only related to the state and decision-making of the previous moment. Therefore, the target optimization problem can be expressed as a discrete-time MDP. As shown in Fig.2, the MDP can be described by$(S_{t-1}, A_{t-1}, r_{t-1}, S_t)$, where S_{t-1} represents the state matrix of the environment at $t-1$, A_{t-1} represents the action vector at $t-1$, and r_t represents the reward value at t. The agent maximizes the reward for the action by learning how to map the environmental state to the

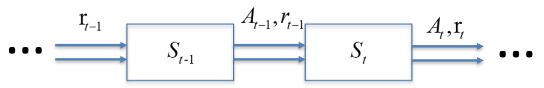

Fig. 2. MDP model.

optimal action through trial and error experience. The specific definitions as follows

State space: The state information describes the environment in which the agent is located, and the agent performs corresponding actions according to the state information. The state information of the multi-beam satellite system is determined by the information recorded in the satellite buffer, and update the information after execution. To this end, we define the state as

$$S_t = \{\Upsilon_t^{c_1}, ..., \Upsilon_t^{c_n}\}^\top = \begin{pmatrix} \phi_{t,l_{th+1}}^{c_1} & \cdots & \phi_{t,0}^{c_1} \\ \phi_{t,l_{th+1}}^{c_2} & \cdots & \phi_{t,0}^{c_2} \\ \cdots & \cdots & \cdots & \cdots \\ \phi_{t,l_{th+1}}^{c_n} & \cdots & \phi_{t,0}^{c_n} \end{pmatrix}. \quad (8)$$

Action space: Since our goal is to jointly optimize beam-hopping scheduling and coverage control strategies, our action space needs to include the above two parameters. Therefore, we map the one-dimensional actions output by the DeepBeam resource management module to actual two-dimensional parameters and it's described as

$$a_k = \left(n_t, \chi_t^k\right), A_t = \left(a_1, a_2, ..., a_K\right), \quad (9)$$

where n_t represents the central cell ID of beam selection k, and χ_t^k represents the coverage radius of beam k. If the satellite ground station of a cell is located in the beam k that is centered on n_t and the radius is χ_t^k, we consider the cell to be included in the C_k cell cluster. For simplicity, we assume that the coverage radius of the beam is a discrete variable.

Reward function: In DeepBeam, the goal is to maximize the throughput of the system. Therefore, we define the reward function as follows

$$r = \sum_{n=1}^{N} \wp_t^{c_n}. \quad (10)$$

B. The Architecture of DeepBeam Algorithm

This section details that the working principle of the DeepBeam algorithm.

For the centralized DRL method with satellite as agent proposed in [8], it becomes difficult to implement as the number of beams and cells increases. On the one hand, the action space of beam-hopping scheduling will grow exponentially as the beam increases, then there are $C_N^K = \frac{N!}{K!(N-K)!}$ kinds of options [9]. Other hands, assuming that the satellite has K beams, each beam with D optional size, there are K^D available schemes. Therefore, the size of the entire action space is $C_N^K \times K^D$, which is curse for the DRL algorithm.

To avoid the above problems, the DeepBeam algorithm uses beams as agents. During training, a beam is used as an independent agent. After training, the model is applied to the MBS scenario. For a single beam, there are only $N \times D$ kinds of service schemes to choose from, which effectively avoids the Curse of Dimensionality brought by satellites as agents, and greatly reduces the training difficulty of the algorithm.

Fig.1 shows three modules, namely the MBS environment, the DeepBeam execution and update process, and the state reconstruction process. First, the state information is reconstructed, and the communication demands of each cell to be served within the maximum tolerable delay in the multi-beam satellite buffer are mapped into a multi-dimensional state matrix. Second, input the state matrix S_t into DeepBeam to get the action vector $A_t = (a_1, a_2, ..., a_K)$, and the neural network parameters are updated. Finally, the action vector A_t is returned to the MBS environment for execution, the state matrix S_{t+1} of the next moment is obtained. The decision-making process of DeepBeam algorithm and the updating process of neural network parameters are shown in Algorithm 1. In DeepBeam, although only one agent is trained, multiple beams on the satellite share the algorithm model of one agent in a polling manner to achieve the purpose of simultaneously controlling multiple beams on the satellite. In the decision-making process, because DeepBeam needs to provide coverage strategies for multiple beams in a round-robin manner, which requires a pseudo-update of the status of each beam after the decision is made (assuming that the satellite has executed a_k, and updated the state matrix $S_{t,k}$ to $S_{t,k+1}$, get reward $r_{t,k}$), if the state pseudo-update is not performed, multiple agents will output the same policy. The trained DeepBeam algorithm model can be deployed on a ground satellite operations control center (GSCC).

Algorithm 1 The DeepBeam Algorithm

```
Input: Initialize Q-network and target network of each agent with random parameter $\theta$.
Initialize target network $Q^-$ with weights $\theta^- \leftarrow \theta$.
Initialize replay buffer $\Re$, exploration parameter $\epsilon$, minibatch size $B$.
 1: Receive initial observation state $s_t$
 2: for step $t = 1$ to $step_{max}$ do
 3:     Agent observe the environment state $s_{t,1} = s_t$.
 4:     for beam $k = 1$ to $K$ do
 5:         Agent input $s_{t,k}$ into the Q-network and get the action $a_{t,k}$.
 6:         Execute action $a_{t,k}$ and state Pseudo-updates and get Pseudo-state $s_{t,k+1}$.
 7:         Agent receive the reward $r_{t,k}$.
 8:         Agent stores the tuple $(s_{t,k}, a_{t,k}, r_{t,k}, s_{t,k+1})$ into $\Re$.
 9:         Agent samples a mini-batch of $(s_{i,j}, a_{i,j}, r_{i,j}, s_{i,j+1})$ from $\Re$ and calculates the loss $L(\theta)$.
10:         Train the Q-network.
11:     end for
12:     Execute action $A_t$ and get next real state $s_{t+1}$.
13:     Update the target network parameter $\theta^-$ of agent every $G$ steps.
14: end for
```

C. The Structure and Training of the Q-network

Since the action space defined in Eq.(9) is discrete, we use the deep Q-network learning method to let the agent learn th e policy. The Q-network is an action-value network that evaluates the expected cumulative reward of certain behavior in a certain state [12]. Unlike traditional reinforcement learning (e.g., Q-learning), our action-value function is implemented through a neural network rather than a Q-table. Here, Since the state information is the communication demand distribution of each cell, and CNN is usually used for signals with a certain spatial structure, so, we adopt CNN as the basic neural network architecture. We train a deep convolutional neural network

(CNN) to obtain the optimal action-value function and then obtain the optimal policy accordingly [9].

$$Q^*(s,a)=\max_{\pi}\mathbb{E}\left[r_t+\gamma r_{t+1}+\cdots \mid s_t=s,a_t=a\right], \quad (11)$$

where γ is the discount factor.

It is well known that the training results of deep reinforcement learning algorithms are overestimated or unstable or even divergent when CNN is used to approximate the Q-value function without resorting to other tricks. To avoid the overestimation problem, a dual network technique (policy network and target network) is adopted in our DeepBeam algorithm [7]. In order to stabilize the training results, we add memory replay technology to the DeepBeam algorithm to improve the stability of the Q-network. Record the training experience in the replay buffer and update the buffer. During the training process, the minibatch is randomly selected from the replay buffer to calculate the target value based on the Bellman equation [13]. The detail is as follows

$$y_t=r_t+\gamma\cdot\max Q^-\left(s_{t+1},a;\theta^-\right), \quad (12)$$

where θ^- is the parameter of the target network and updated every G steps using the parameter θ of the policy network Q, and remains fixed between individual updates. The loss value $L(\theta)$ of the current Q-network at t can be obtained by the target value in Q^* as follows [14]

$$L_t\left(\theta_t\right)=\mathbb{E}_{\left(s_t,a_t,r_t,s_{t+1}\right)\sim U(\Re)}\left(y_t-Q\left(s,a;\theta_t\right)\right)^2, \quad (13)$$

where θ_t is the policy network parameter at t. We train the Q-network by minimizing the loss value $L(\theta)$.

IV. Evaluation

A. Simulation Parameters

In this section, we present simulation experiments to evaluate our proposed algorithm and analyze the simulation results. Simulations are performed on a Python 3.6 platform. All simulations are performed on an Intel(R) Core(TM) i5-1135G7, 8G of RAM, and Intel(R) UHD Graphics630 (used in the reinforcement learning training phase). The simulation scenario is a GEO multi-beam satellite operating in the Ka frequency band and running in the geosynchronous orbit. The main parameters are given according to the GEO mobile wireless interface specification. Table.I summarizes the simulation parameters, some of these are based on [8]. The communication demands of each cell follow a Poisson distribution. Due to the uneven geographical distribution of users, the service demands arrival rate of each cell is different. In order to reflect the time-varying characteristics of communication demands, the service arrival rate of each cell changes once in each time slot.

B. Performance Comparison

To verify the impact of the proposed DeepBeam algorithm on the performance of MBS systems, we compare the proposed method with different schemes as follows

- The Random Beam Hopping and Coverage (Random): Randomly determine the center cell and beam size covered by the beam.
- The Greedy-Based Beam Hopping and Coverage (Greedy): According to [6], select the 7 cells with the largest communication demands as the central cells, and randomly determine the beam size.
- The Genetic Algorithm Beam Hopping and Coverage (Genetic): According to [4], the satellite makes the beam plan and coverage plan with the genetic algorithm. The population is set as $G=60$ and the number of generation is set as $P=600$.
- The DRL Algorithm Beam Hopping Without Coverage (DRL-BH-X): According to [8], the satellite determines the beam hopping strategy based on DRL, but does not include coverage control. The optimization goal is to maximize throughput.
- The MADDPG Algorithm Beam Hopping and Coverage (MADDPG): According to [2], the satellite determines beam hopping strategy and coverage control strategy based on MADDPG algorithm. The optimization goal is to maximize throughput.

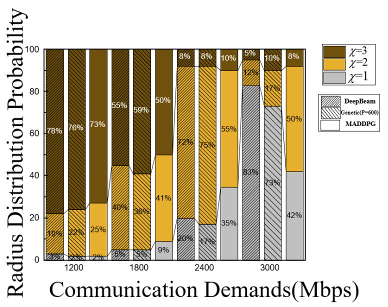

Fig. 7. The Beam radius distribution.

TABLE I. SIMULATION PARAMETERS

PARAMETERS	VALUES
SCENARIO PARAMETERS	
Satellite altitude	35786 km
K_a Band f_c	20 GHz
Total available bandwidth B_{tot}	500 MHz
Bandwidth of a chunk B_c	71 MHz
Total satellite power P_{tot}	34.5 dBW
Beam power	26 dBW
Number of beams K	7
Number of cells N	30
Maximum transmit antenna gain	40.3dBi
Terminal antenna aperture d	0.25m
Free space loss L_f	209.6 dB
Maximum receiving antenna gain G_r	31.6 dBi
Time slot duration	2ms
Maximum tolerated delay l_{th}	60ms
ALGORITHM PARAMETERS	
Replay memory capacity $\Re$	100000
Minibatch size B	128
Learning rate lr	0.0001
Target network update frequency G	100
discount factor γ	0.95

The beam coverage effects of different radii are shown in Fig.3, where the cell cluster covered by beam 1 includes five cells with the smallest beam capacity, the cell cluster covered by beam 2 includes one cell with the largest beam capacity, and beam three covers cell cluster consists of 3 cells.

Fig.4 shows the evolution of rewards during training for the proposed DeepBeam algorithm and four other DRL-based methods. During the training process, the total communication

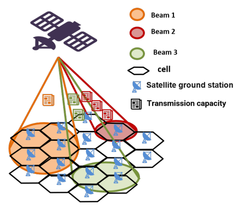

Fig. 3. The MBS system with variable beam coverage radius.

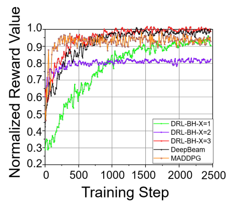

Fig. 4. The normalized reward value.

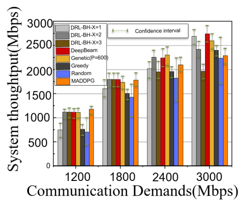

Fig. 5. The system throughput.

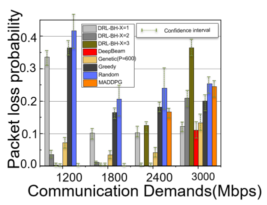

Fig. 6. The packet loss probability.

demand of each cell at each moment is 2400 Mbps. We note that the reward values after convergence between DeepBeam and DRL-BH-$\chi = 2$ are similar because when the total communication demand is 2400 Mbps, the beam radius $\chi = 2$ is the optimal beam coverage scheme.

Fig.5 and Fig.6 show the relationship between throughput and packet loss probability and communication demands for different methods. Since the communication demand of each cell is generated randomly according to Poisson distribution, the test result is not a fixed value. In order to make the experimental result more referrable, we conduct 500 tests on each algorithm and take the statistical average of the test result as the final result. It can be observed that DeepBeam has a significant performance improvement compared to the three DRL-based algorithms without considering beam coverage. Specifically, the DeepBeam algorithm is more obvious in reducing the packet loss probility. This is because of the unique ability of the DeepBeam algorithm to adapt the beam coverage radius. In addition, our proposed DeepBeam has 20%-34% higher throughput and 10%-30% lower packet loss probability compared to Random, Greedy, and Genetic. We note that the proposed DeepBeam algorithm significantly outperforms the MADDPG algorithm when the high demands scenario.

Fig.7 shows the radius distribution of three algorithms considering the beam coverage radius under four different communication demands. We notice that all three algorithms tend to cover more cells when the total communication demands is relatively small (i.e., $\chi = 3$). However, when the total communication demands is large, the strategy based on DeepBeam is more inclined to cover fewer cells (i.e., $\chi = 1$), which makes the transmission capacity of a single beam larger. Its performance is shown in Fig.5 and Fig.6. The DeepBeam algorithm has better performance in the communication scenario where the communication demand is 3000Mbps.

V. Conclusion

This paper proposes a joint optimization algorithm of beam hopping and beam coverage for MBS. The novelty lies in optimizing the performance of multi-beam satellites from the perspective of beam coverage. The Deepbeam algorithm to consider long-term benefits, and adopts single-model multiplexing technology and state pseudo-update techniques reduce the dimensionality of the search space. The simulation results show that compared with other reinforcement learning algorithms that do not consider beam coverage, this method can always maintain the optimal performance in various scenarios. In addition, compared with other heuristic methods, the policy based on DeepBeam improves the throughput by 20%-34% and reduces the packet loss probability by 10%-30%. When compared with the MADDPG algorithm for multi-agent reinforcement learning, we find that DeepBeam is relatively better suited to scenarios with high communication demands.

References

[1] G. Cocco, T. de Cola, M. Angelone, Z. Katona, and S. Erl, "Radio resource management optimization of flexible satellite payloads for dvb-s2 systems," *IEEE Transactions on Broadcasting*, vol. 64, no. 2, pp. 266–280, 2018.

[2] X. Liao, X. Hu, Z. Liu, S. Ma, L. Xu, X. Li, W. Wang, and F. M. Ghannouchi, "Distributed intelligence: A verification for multi-agent drl-based multibeam satellite resource allocation," *IEEE Communications Letters*, vol. 24, no. 12, pp. 2785–2789, 2020.

[3] S. Kisseleff, E. Lagunas, T. S. Abdu, S. Chatzinotas, and B. Ottersten, "Radio resource management techniques for multibeam satellite systems," *IEEE Communications Letters*, vol. 25, no. 8, pp. 2448–2452, 2021.

[4] L. Wang, X. Hu, S. Ma, S. Xu, and W. Wang, "Dynamic beam hopping of multi-beam satellite based on genetic algorithm," in *2020 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking (ISPA/BDCloud/SocialCom/SustainCom)*, pp. 1364–1370. IEEE, 2020.

[5] D. Shi, F. Liu, and T. Zhang, "Resource allocation in beam hopping communication satellite system," in *2020 International Wireless Communications and Mobile Computing (IWCMC)*, pp. 280–284, 2020.

[6] F. Tian, L. Huang, G. Liang, X. Jiang, S. Sun, and J. Ma, "An efficient resource allocation mechanism for beam-hopping based leo satellite communication system," in *2019 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB)*, pp. 1–5. IEEE, 2019.

[7] X. Hu, Y. Zhang, X. Liao, Z. Liu, W. Wang, and F. M. Ghannouchi, "Dynamic beam hopping method based on multi-objective deep reinforcement learning for next generation satellite broadband systems," *IEEE Transactions on Broadcasting*, vol. 66, no. 3, pp. 630–646, 2020.

[8] X. Hu, S. Liu, Y. Wang, L. Xu, Y. Zhang, C. Wang, and W. Wang, "Deep reinforcement learning-based beam hopping algorithm in multibeam satellite systems," *IET Communications*, vol. 13, no. 16, pp. 2485–2491, 2019.

[9] Z. Lin, Z. Ni, L. Kuang, C. Jiang, and Z. Huang, "Dynamic beam pattern and bandwidth allocation based on multi-agent deep reinforcement learning for beam hopping satellite systems," *IEEE Transactions on Vehicular Technology*, 2022.

[10] T. Zhang, L. Zhang, and D. Shi, "Resource allocation in beam hopping communication system," in *2018 IEEE/AIAA 37th Digital Avionics Systems Conference (DASC)*, pp. 1–5. IEEE, 2018.

[11] C. Zhang, J. Jin, H. Zhang, and T. Li, "Spectral coexistence between leo and geo satellites by optimizing direction normal of phased array antennas," *China communications*, vol. 15, no. 6, pp. 18–27, 2018.

[12] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[13] Y. Ran, X. Zhou, H. Hu, and Y. Wen, "Optimizing data centre energy efficiency via event-driven deep reinforcement learning," *IEEE Transactions on Services Computing*, pp. 1–1, 2022.

[14] Y. Ran, H. Hu, Y. Wen, and X. Zhou, "Optimizing energy efficiency for data center via parameterized deep reinforcement learning," *IEEE Transactions on Services Computing*, pp. 1–14, 2022.