Trajectory Planning using Q-learning for Multi-UAV Enabled Up-Link NOMA

Alexandros E. Tzikas

I. Problem Description

A novel online trajectory planning algorithm for a two-dimensional grid space, two-UAV, with stochastic user presence up-link NOMA problem is presented. A Markov Decision Process (MDP) formulation is developed that enables the application of the well-known Q-learning algorithm for online learning of the optimal policy. The proposed system is able to satisfy the user demands by finding the optimal policy; the optimal trajectory in the grid space that leads to the maximum supplied sum-rate to the users, while also keeping the two UAVs in a safe distance from each other.

The following assumptions are made:

- A two-dimensional grid space is considered with each cell able to host a UAV and/or a user in its center point. The UAV is assumed to be flying above the center point of the cell.
- Time is discrete and each UAV can make one move in each time slot (up, down, right, left).

II. Centralized Implementation

A. Reinforcement Learning Problem Formulation

The problem formulation is based on Finite Markov Decision Processes (FMDP), as shown in [1]. To define an FMDP reinforcement learning problem we need to define a tuple $\{S, A, R, p\}$, where S is the state space with finite number of elements, A is the action space with finite number of elements and R is the reward space with finite number of elements. We assume that from each state, the action space is the same. The probability function p represents the dynamics of the environment and is defined as:

$$p\left(s', r|s, a\right) \ = \ Pr\left\{S_{t+1} = s', R_{t+1} = r \mid S_t = s, A_t = a\right\}. \tag{1}$$

The goal is to find the policy $\pi\left(a|s\right)$, the action a to be taken at state s, that leads to the highest expected cumulative reward. Due to the existence of two UAVs we formulate the problem as follows in a centralized manner (a central processor determines the behavior of both UAVs and knows the position of the UAVs and the demands at each time).

We consider a two-dimensional grid

$$G \ = \ \{S_i \ = \ (x_i, y_i), \forall x_i \in \{1, \cdots, N_x\}, \ \forall y_i \in \{1, \cdots, N_y\}\}, \tag{2}$$

where N_x, N_y represent the total number of cells horizontally and vertically. In other words, G consists of a certain number of cells.

Due to the coexistence of the two UAVs, we define the state space as:

$$S \ = \ \{\overbrace{(s_1, \ s_2)}^{s}, \ \forall \ s_1, \ s_2 \in G\}, \tag{3}$$

where s_1, s_2 are the cells in the grid of UAVs 1 and 2 respectively.

Correspondingly, we define the action space as:

$$A \ = \ \{\overbrace{(a_1, \ a_2)}^{a}, \ \forall \ a_1, \ a_2 \ \in \ \overbrace{\{1, \ 2, \ 3, \ 4\}}^{\{\text{left, up, right, down}\}} \ \}, \tag{4}$$

where a_1, a_2 are the actions of UAVs 1 and 2 respectively.

The dynamics of the system can therefore be expressed as:

$$p\left(S_{t+1} = \overbrace{(s_1', \ s_2')}^{\text{next state } s'}, \ R_{t+1} = r \mid S_t = \overbrace{(s_1, \ s_2)}^{\text{current state } s}, \ a_t = \overbrace{(a_1, \ a_2)}^{\text{current action } a}\right). \tag{5}$$

It is important that the same dynamics equation holds $\forall t$.

The Q-learning algorithm will search for the optimal deterministic policy:

$$\pi^*\left(\overbrace{(a_1, \ a_2)}^{\text{current action } a} \ \mid \ \overbrace{(s_1, \ s_2)}^{\text{current state } s}\right), \tag{6}$$

by following an ϵ-greedy policy and completing episodes of a certain duration.

Finally, the reward for the transition to a state $(s_1,\ s_2)$ is defined as follows, based on [2]:

$$r \ = \ \begin{cases} \log_2\left(1+\frac{P}{\sigma^2}\sum_{k_1\in K_1} h_{k_1}\right) + \log_2\left(1+\frac{P}{\sigma^2}\sum_{k_2\in K_2} h_{k_2}\right), & \text{if } s_1 \ \neq \ s_2 \\ -50, & \text{otherwise} \end{cases}, \tag{7}$$

where h_k denotes the power gain from user k to the UAV that provides its service at the given time slot (state $(s_1,\ s_2)$)

$$h_k \ = \ \frac{1}{d^2}, \tag{8}$$

where d is the distance from the user to the UAV that the user communicates with. K_1, K_2 denote the sets of users serviced by UAVs $1, 2$ respectively at that time slot. A user is assigned to the UAV to which the user's power gain is higher, because:

$$\log_2(a+b) + \log_2(c) > \log_2(a) + \log_2(c+d), \ \ a,b,c,d > 0, \ \ d < b. \tag{9}$$

To encourage wide area coverage and collision safety between the UAVs, a transition for both UAVs to the same cell is discouraged via a negative reward. It is important to note that no interference between the two UAVs is considered in the reward function.

B. Simulations

1) Proposed Method: We assume a user stochastic presence, $Pr\{\text{user in cell } i \text{ at time slot t}\}$, $\forall t$, where at each time slot a user arises in cell i with the given probability. This means that the same eq. (5) holds $\forall t$. In this simulation, we consider a rectangular 20×10 grid with two sub-areas where users might arise. These areas are (4:6)×(3:7) and (14:16)×(3:7). In each time slot, at each of the cells of the two aforementioned areas, a user might arise according to some probability (here 0.7). In the learning process of the algorithm, no fixed initial state is assumed, meaning that the UAVs learn to behave optimally given any initial positions on the grid. It is logical to assume that the optimal policy learned by the system should guide one UAV on top of each of the two user clusters. The UAVs should then stay relatively static. This is exactly the outcome. After the Q-learning phase, in which the UAVs attempt to behave optimally (ϵ-greedy behavioral policy) but are still learning the optimal Q table, a testing phase was introduced. In an effort to truly challenge the performance of the learning phase, in the testing phase both UAVs where initially placed on the right edge of the grid. If the system indeed managed to learn the optimal policy, one UAV should move to the left cluster, while the other should stay with the right in the test. This is exactly what happened in the simulation, indicating the effectiveness of the proposed algorithm. As seen in Fig. 1, UAV 1 moves towards the first cluster (state 45 i.e., cell (5, 5)) and UAV 2 towards the other cluster (state 145 i.e., cell (15,5)). It is also note-worthy that the UAVs remain at the center point of the two areas, as expected, to increase the rate by decreasing the distance to the users. The small variation in state only occurs because the UAVs are not allowed to stay in the same cell (the allowed actions are up, down, left, right). In Fig. 2, the cumulative reward achieved via the ϵ-greedy behavioral policy of each episode is seen. In each episode, this includes the sum-rate of each time slot and it might include the negative reward for the case of the two UAVs coexisting in a cell at a given time slot. In the simulation it is assumed that $P = 23\ dBm, \sigma^2 = -80\ dBm$.

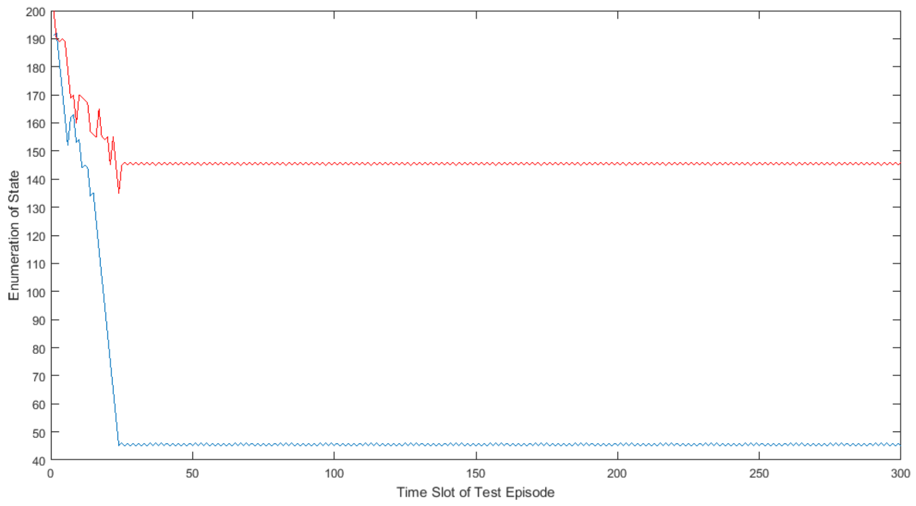

Fig. 1: Followed Path after Learning for Centralized Method.

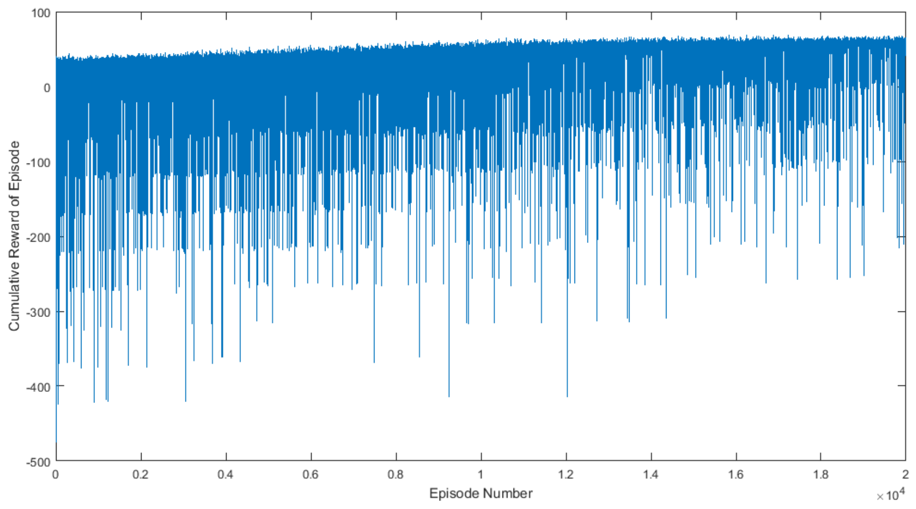

Fig. 2: Cumulative Reward per Episode for Centralized Method.

2) Conventional Optimization Approach: In the scenario discussed in this section II-B, the aforementioned proposed system will be compared with a classic optimization approach. The results will show that the Q-learning method achieved identical instantaneous sum-rate and, given its online learning capability, provides an attractive alternative to the classical optimization approach that requires, as will be shown, knowledge of $Pr\{\text{user in cell } i \text{ at time slot t}\},\ \forall t$.

Provided knowledge of the existence of users in two distinctly shaped areas in the grid only (clusters) and given the existence of two UAVs, we assume that one UAV will be assigned to serve each cluster. Any other scheme, will have a decreased performance, because of greater distances between the user and the agent providing the service. Therefore, the optimization problem to solve is:

$$\max_{S_i \in G} \log_2 \left(1 + \frac{P}{\sigma^2} \sum_{k \in K_i} R_k^i \right),\ i \in \{1,2\} \tag{10}$$

for each drone i. However, R_k^i are random variables with the probability distribution function:

$$Pr\{R_k^i\} = \begin{cases} 0.7, & \text{if } R_k^i = \frac{1}{(\chi_k - \chi_i)^2 + (\psi_k - \psi_i)^2 + 100^2} \\ 0.3, & R_k^i = 0 \end{cases}, \tag{11}$$

where (χ_i, ψ_i) is the position of the i-th UAV in the grid (corresponding to state $S_i = (x_i, y_i)$) and (χ_k, ψ_k) is the position of the user k. The position coordinate χ, ψ is derived from the cell enumeration coordinate x, y respectively, by taking into account the size of the cells. We also assume a UAV flying altitude of $100m$.

This leads to a maximization in terms of expected value. Because $\log_2$ is a monotonous function, we solve the following problem instead:

$$\max_{S_i \in G} E\left[\sum_{k \in K_i} R_k^i \right],\ i \in \{1,2\}, \tag{12}$$

which is equivalent to:

$$\max_{S_i \in G} \sum_{k \in K_i} E\left[R_k^i\right],\ i \in \{1,2\}. \tag{13}$$

Therefore, we solve:

$$\max_{S_i \in G} \sum_{k \in K_i} \frac{0.7}{(\chi_k - \chi_i)^2 + (\psi_k - \psi_i)^2 + 100^2},\ i \in \{1,2\}. \tag{14}$$

It is evident that the optimization problem assumes knowledge of the 0.7 probability of appearance mentioned above.

We remind the reader that, in the case of the classic optimization formulation, the two sets of users serviced by the two UAVs are fixed and identical to the two user clusters:

$$K_1 = \{(x_i, y_i),\ x_i \in \{4,5,6\}, y_i \in \{3,4,5,6,7\}\}, \tag{15}$$

$$K_2 = \{(x_i, y_i),\ x_i \in \{14,15,16\}, y_i \in \{3,4,5,6,7\}\}. \tag{16}$$

The optimization of eq. (14) is easily solvable due to the finite amount of states-positions that need to be tested for each UAV (200 cells in this simulation).

3) Results: The results of Fig. 3 show that the two methods presented above have the same performance, when it comes to the sum-rate achieved in each time-slot for the testing scenario mentioned above. The lower sum-rate achieved via the proposed system in the initial time slots is attributed to the initialization of the UAVs' positions in the right side of the grid, i.e., the UAVs must move towards the two clusters.

The classical optimization approach has the requirement of prior knowledge of $Pr\{\text{user in cell } i \text{ at time slot t}\}, \ \forall t$, and prior knowledge of the existence and shape of the two clusters. These requirements do not extend to the Q-learning approach.

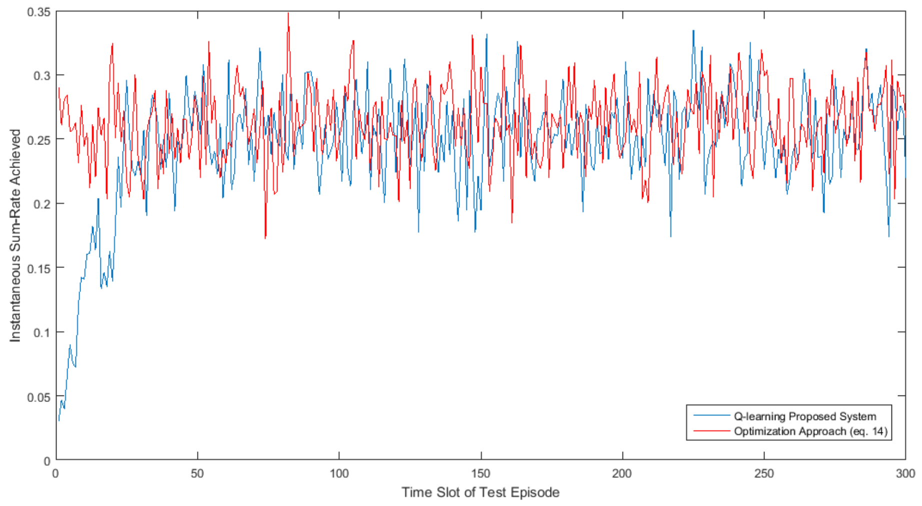

Fig. 3: Instantaneous Sum-Rate of Each Time Slot.

C. Generalization

The algorithm's performance and the easy problem formulation in the two-UAV case indicate that the idea above can be applied in scenarios with multiple UAVs. Indeed, the problem formulation will just be a generalization of the analysis in Section II-A. However, the size increase of the state and action spaces might require the implementation of a slightly different methodology (Ch. 9, 10, and 11 in [1]). In addition, the proposed system is applicable in various different scenarios, not only in cases where the users form distinct clusters and the optimal trajectory turns into a quasi-static behavior. The scenario with the two clusters only serves as an initial validation of the proposed system.

III. Decentralized Implementation

In this section, we will demonstrate a decentralized trajectory planning approach for a multi-UAV system using reinforcement learning and Q-learning in particular. We will present an algorithmic implementation for decentralized path-planning, for which the only information exchange requirement between the UAVs is the following:

- Each UAV has knowledge of the position-state of all other UAVs (each UAV knows where every UAV will be in the new state-next time slot).

The idea is that each UAV applies Q-learning on its own Q table. Each UAV has knowledge only of its own sum-rate (the sum-rate for the users it serves) and knowledge of the positions of the other UAVs (therefore it can determine the group of users it will serve by determining if they are closer to itself or the other UAVs, eq. (9)).

Therefore the formulation that each UAV i "sees" is the following:

$$p\left(s_{t+1} = s_i', \ R_{t+1} = r \mid s_t = s_i, \ a_t = a_i\right), \tag{17}$$

with $s_i', s_i \in G$ and $a_i \in \overbrace{\{1, \ 2, \ 3, \ 4\}}^{\{\text{left, up, right, down}\}}$.

Each UAV tries to find its own optimal policy

$$\pi_i^*\left(\overbrace{a_i}^{\text{current action of UAV } i} \mid \overbrace{s_i}^{\text{current state of UAV } i}\right), \tag{18}$$

by updating its Q table

$$Q_i\left(s_i, a_i\right). \tag{19}$$

The key part in this approach is integrating the knowledge of the state-position of the other UAVs in the learning process of each UAV. In other words, determine how each UAV, knowing only the sum-rate it provides, will have a sense of its position's optimality with regard to the system's total achieved sum-rate (the sum of sum-rates of all UAVs). This can be done by using a slight variation of the following reward function at each UAV i:

$$r_i = \begin{cases} \log_2\left(1 + \frac{P}{\sigma^2}\sum_{k \in K_i} h_k\right), & \text{if } s_i \neq s_j, \ \forall j \neq i \\ -5, & \text{otherwise} \end{cases}. \tag{20}$$

The change from -50 in eq. (7) to -5 in eq. (20) is so that the penalty for an unsafe move (that is deemed unsafe only in this realization) does not completely discard the action as unwanted due to a great negative reward that cannot be compensated in future episodes.

Instead of using eq. (20), in order to integrate the knowledge of the other UAVs' positions, we scale this reward received at UAV i when the UAVs have moved to positions $\{s_1, ..., s_N\}$, with the frequency-probability of the other UAVs occupying their positions given that UAV i is at s_i. This is done independently at the processor of each UAV. Each UAV tends to optimize its own sum-rate and, given that it knows the sum-rate it provides, when the UAVs have positions $\{s_1, ..., s_N\}$, it understands how probable this sum-rate is (how probable it is for the other UAVs to have their positions when it is at s_i). In other words, it knows how probable it is to get this sum-rate (serve this group of users due to the position of the other UAVs) if it is at a specific position, by scaling with the probability of the other UAVs having their current positions. Specifically, if it achieves a high sum-rate by moving to a position at one realization of the game, this does not necessarily mean that this move is indeed the best, because it might be improbable that from this position it will serve the users it now serves (since the current position of the other UAVs might be unlikely).

The algorithm is seen below:

1) For each UAV i, initialize its Q table $Q_i(s_i, \ a_i)$ and its frequency table $F_i(s_i, \overline{s})$, where $\overline{s}$ denotes the states of the other UAVs. F_i serves as the probability scaling factor for the reward of UAV i at each iteration.
2) For each episode:
   a) Select an initial state for each drone s_i.
   b) Find the possible actions of UAV i from its state s_i.
   c) For each time slot of the episode:
      i) For every UAV i which is at s_i, using its possible actions, select its action a_i based on the ϵ-greedy policy.
      ii) For every UAV i, move the UAV based on the selected action and find the sum-rate r_i it offers in the next state s'_i. Since each UAV knows the positions of all other UAVs, it can determine the group of users it serves.
      iii) For each UAV i, update its F_i table by adding one unit to the cell that corresponds to the next state of the environment (the next positions-states of all UAVs, $[s'_i, \overline{s'}]$).
      iv) For each UAV i, scale its sum-rate at s'_i in this episode by the likelihood of it actually happening:

$$r_i \leftarrow \frac{F_i(s'_i, \overline{s'})}{\sum_{\overline{s'}} F_i(s'_i, \overline{s'})} r_i \tag{21}$$

      v) Apply the Q-learning iteration

$$Q(s_i, a_i) \leftarrow Q(s_i, a_i) + \alpha\left(r_i + \gamma \ max_{a'_i} Q(s'_i, a'_i) - Q(s_i, a_i)\right). \tag{22}$$

      vi) For each UAV i, update its current state by its next state:

$$s_i \leftarrow s'_i. \tag{23}$$

      vii) For each UAV i, find the possible actions from its current state.

A. Simulations

We will use the same testing scenario as in Section II-B to test the performance of the decentralized approach. In Fig. 5, the cumulative reward achieved via the ϵ-greedy behavioral policy in each episode is seen. In each episode, this includes the sum-rate of each time slot and it might include the negative reward for the case of the two UAVs coexisting in a cell at a given time slot. In this simulation it is assumed that $P = 23 \ dBm, \sigma^2 = -80 \ dBm$, as before. As seen in Fig. 4, UAV 1 moves towards the first cluster (state 45 i.e., cell (5, 5)) and UAV 2 towards the other cluster (state 155 i.e., cell (16,5)), which is almost identical to what happened in the systems simulated in Section II-B. In other runs, the algorithm exhibited convergence to states 45 and 145.

Finally, in Fig. 6, the instantaneous (per time-slot) sum-rate of the three methods (Sections II-B1, II-B2, III) are shown for the testing scenario of Section II-B. The decentralized method shows almost equivalent performance with the centralized one for the same number of training episodes. In other runs, the variation in the path was smaller and the sum-rate slightly elevated. This, combined with the fact that the decentralized algorithm enables the UAVs to learn independently with minimal information exchange, while not requiring a central processing unit, indicates that it is an attractive alternative.

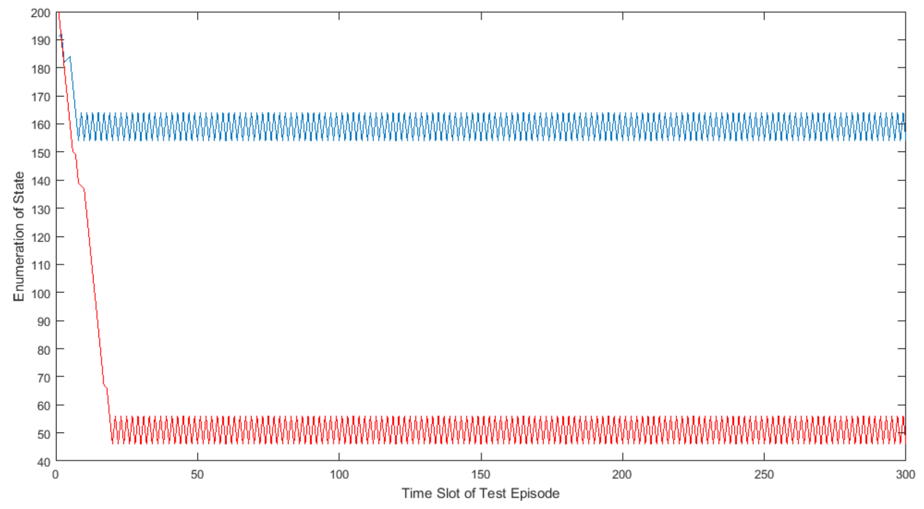

Fig. 4: Followed Path after Learning for Decentralized Method.

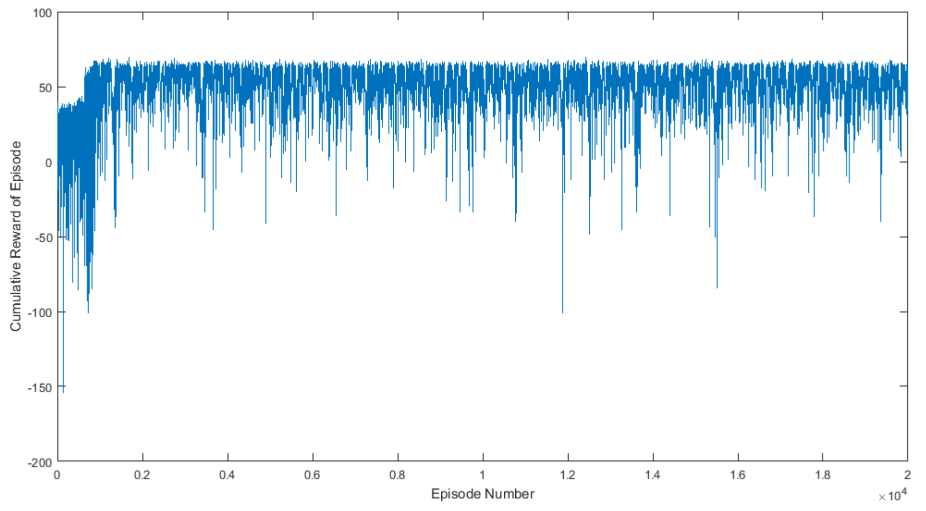

Fig. 5: Cumulative Reward per Episode for Decentralized Method.

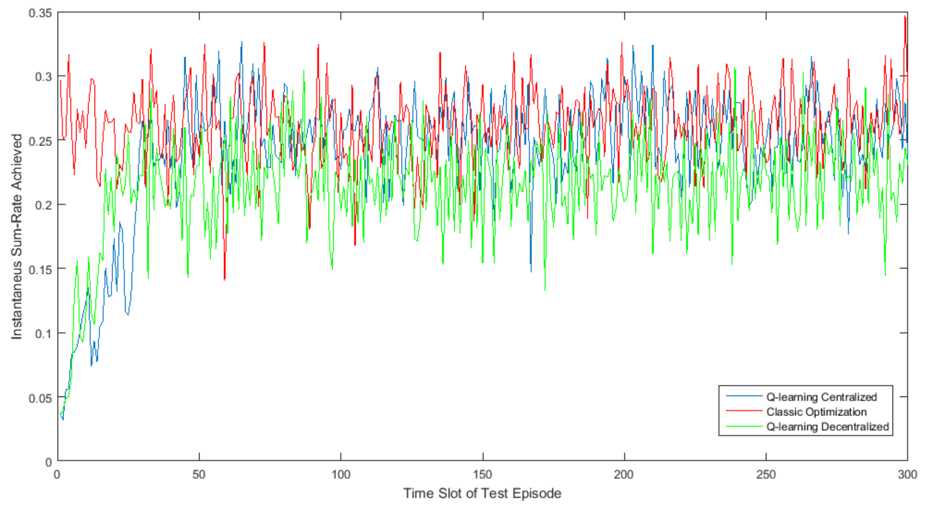

Fig. 6: Instantaneous Sum-Rate of Each Time Slot for All Methods.

References

[1] R. Sutton and A. Barto, *Reinforcement Learning: An Introduction*, 2nd ed. Cambridge, Massachusetts, USA: The MIT Press, 2018.

[2] Y. Huang, X. Mo, J. Xu, L. Qiu, and Y. Zeng, “Online Maneuver Design for UAV-Enabled NOMA Systems via Reinforcement Learning,” in *2020 IEEE Wireless Communications and Networking Conference (WCNC)*, 2020, pp. 1–6.