DOI: 10.17516/1997-1397-2022-15-4-431-443
УДК 623.618, 004.85

Intellectual Ways of Solving the Problem of Constructing the Optimal Route of an Unmanned Aerial Vehicle in the Conditions of Counteraction

Igor N. Ischuk*
Siberian Federal University
Krasnoyarsk, Russian Federation
Bogdan K. Telnykh†
Military Training and Research Center of the Air Force
«Air Force Academy ft. Professor N.E. Zhukovsky and Y.A. Gagarin»
Voronezh, Russian Federation
Valeriy N. Tyapkin‡
Nikolay S. Kremez§
Siberian Federal University
Krasnoyarsk, Russian Federation

Received 05.02.2022, received in revised form 09.04.2022, accepted 12.05.2022

Abstract. Traditional and modern algorithms for solving the problem of planning the optimal route of an unmanned aerial vehicle under the influence of low-altitude air defense systems is presented in the paper. The principles of the methods, as well as, the tools used in them are described. Classical approaches of reinforcement learning and its modification using artificial neural networks are considered. The proposed algorithms are implemented and simulation with the use of these algorithms is carried out. A comparative analysis of the results is performed and conclusions about the effectiveness of the algorithms are presented.

Keywords: unmanned aerial vehicle, artificial neural network, information technologies, intelligent agent, reinforcement learning.

Citation: I.N. Ischuk, B.K. Telnykh, V.N. Tyapkin, N.S. Kremez, Intellectual Ways of Solving the Problem of Constructing the Optimal Route of an Unmanned Aerial Vehicle in the Conditions of Counteraction, J. Sib. Fed. Univ. Math. Phys., 2022, 15(4), 431–443.
DOI: 10.17516/1997-1397-2022-15-4-431-443.

Modern unmanned aerial vehicles (UAVs) are multifunctional high-tech intelligent carriers that are currently used in various fields. One of the ways of effective use of UAV's is to apply of navigation support for autonomous flight using elements of artificial intelligence [1]. At present, many studies have been devoted to solving the problem of planning optimal UAV routes but most of them only reflect ways to overcome passive natural and anthropogenic obstacles. However, the appearance of radio suppression devices and modern air defense systems («active obstacles») significantly affects the trajectory of the flight of UAVs [2, 3]. Therefore, these obstacles should

*boerby76@mail.ru
†bogdanfm0508@yandex.ru
‡tyapkin58@mail.ru https://orcid.org/0000-0002-3087-8653
§nkremez@sfu-kras.ru

also be taken into account when planning flight tasks. To solve this problem, some algorithms for constructing the optimal UAV route in the conditions of «active obstacles» are considered and analyzed.

1. The traditional UAV flight route planning algorithm

Let us consider a traditional algorithm for constructing a route. The algorithm is based on a random generator of the UAV flight direction vector with an estimate of the shortest path using a Euclidean metric. Then the distance between points is

$$d(x,y) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + \cdots + (x_n - y_n)^2} = \sqrt{\sum_{k=1}^{n} (x_k - y_k)^2}. \qquad (1)$$

Mathematical modeling of the runtime environment and numerical calculations were performed using the Python programming language.

At the first stage of simulation a tactical situational map of the UAV flight area is formed. It is a unit matrix of size 100 X 100 with «active obstacles» highlighted. These units have identical target detection and destruction radii. It is assumed that as soon as the UAV enters the affected area it will be immediately destroyed and the simulation episode is finished.

At the second stage the UAV direction vector is determined. Suppose that in the area of the flight the UAV can move in any direction. There is also a restriction on crossing the map border. In this case, the UAV takes a step back and continues to move in accordance with the value of the random direction vector generator. In total, 4 directions of discrete actions are recorded which are equivalent to moving along the cardinal directions.

At the third stage a direction is generated with an estimate of the choice of the shortest path according to expression (1). Here the UAV route in the specified area can be directly plotted. Fig. 1 shows options for constructing a UAV route using the presented algorithm where yellow and red indicate the detection and destruction zones of air defense.

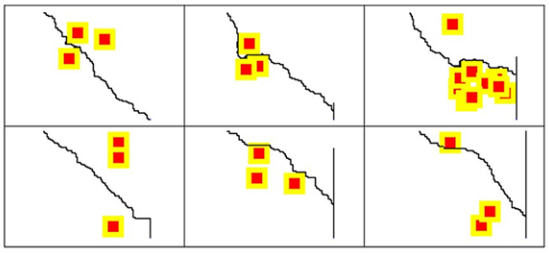

Fig. 1. Variants of UAV flight routes under counteraction conditions. The routes are calculated using the algorithm of a random direction vector generator with an estimate of the shortest distance according to the Euclidean metric

The main advantages of this algorithm are simplicity of its implementation and the small amount of calculations but the direction vector should be generated many times. The flight path of UAV is often chaotic, and the number of steps per episode may be as great as 5000–10000. This problem can be solved by using intelligent methods for constructing the optimal route such as reinforcement learning and artificial neural networks algorithms.

2. Intelligent reinforcement learning methods for constructing the optimal UAV route

2.1. Statement of the reinforcement learning problem

Modern technologies and the growth of computing resources allow one to apply unusual methods to solve navigation problems. It is now possible to take into account factors and conditions that increase the accuracy and reliability of obtained results. One of these approaches is reinforcement learning (RL). There is some environment with a set of states S_n as well as an intelligent agent that can perform certain actions a_n in this environment. It is known that in each state the agent will receive a certain reward $r(S, a)$ for the completed action. The way an agent chooses a direction vector is called a strategy or policy:

$$a = \pi(S), \tag{2}$$

where $\pi(S)$ is the function of selecting the optimal action that depends on the current state.

The aim of the RL is to obtain the maximum reward for the action performed. However, to optimize the path selection one needs to introduce an additional parameter in the form of a discount factor γ in the current step. The factor takes values in the range from 0 to 1. Hence a system of penalties is introduced for an agent that performs many actions to obtain a positive reward. Now the search for the optimal policy can be represented by the following expression [4]:

$$\pi^* = \arg\max \sum_{n \geqslant 0} \gamma^n r_n, \quad \gamma \in [0; 1], \tag{3}$$

where r_n is the nearest discounted reward.

Thus, the task of the RL is to find the optimal strategy for maximizing the future total reward for the minimum number of steps if it is possible.

2.2. The table Q-learning

Let us consider the classic reinforcement learning algorithm — table Q-learning. This approach is also to find the optimal agent policy but to evaluate the future state model the parameter $Q(S, a)$ is introduced. This is the function for evaluating the total discounted reward of a state and action pair. To evaluate a pair (state, action) one needs to know all the ideal further actions. However to know what future actions will be ideal one needs to have precisely calculated state and action values.

There is a contradiction as to how to define the values Q from future values $Q^{'}$. Such relations can be solved using the Bellman equation which states that the maximum future reward from an action a is the current reward plus the maximum future reward in the next step from performing the next action $a^{'}$ [5]. Therefore, taking into account (3), the function can be represented in the following form:

$$Q(S, a) = E[r + \gamma \max_{a'} Q^{'}(S^{'}, a^{'})], \tag{4}$$

where E is a mathematical expectation of the value of the current reward and the discounted total reward.

The table Q-learning implies that function $Q(S, a)$ is represented as a table. Columns and rows of the table correspond to the number of environmental states S_n and the number of actions a_n that agent can perform.

2.3. Implementing table Q-learning to solve the problem of calculating the optimal UAV route under influence of active obstacles

Numerical calculations of the optimal UAV route was carried out using OpenAI Gym–Python library for developing intelligent agents with reinforcement learning.

Why are reinforcement learning algorithms tested in Gym environments? Let us remind that RL does not make many assumptions about the environment; knowledge about it is collected in the process of interaction. In addition, to compare the quality of different algorithms they should be applied in the same standardized environments. Gym is an excellent testing tool because it contains many flexible settings and easy-to-use environments [6].

The algorithm for constructing the optimal UAV route using a table Q-learning consists of the following stages:

Stage 1. Create a simulated UAV flight area in the form of a situation map with the situation plotted and divided into a number of computational areas represented as uniform cells [7] where each corresponds to one move of the agent. The map size is 100x100. As in the traditional approach an intelligent agent that performs the role of a UAV can freely move around the entire area of the environment in four directions. Thus, the number of states of the simulated environment $S = 10000$, and the number of UAV actions $a = 4$.

Stage 2. Defining hyperparameters of training. At this stage values are assigned to training parameters that directly affect the simulation process itself. They are the learning rate α (takes values in the range from 0 to 1), the discount rate of the reward γ , the number of simulation episodes and the maximum number of iterations in one episode. Parameter α is necessary for the UAV to use the knowledge gained in previous episodes to choose more reliable actions. The maximum number of steps per episode is required so that the agent does not get stuck in the loop but the episode may end earlier. The hyperparameter values are shown in Tab. 1.

Table 1. Training Settings

Training coefficient	**Value**
Learning rate, α	0.8
Discount rate, γ	0.99
Number of training episodes	1000000
Maximum number of steps per episode	60000

Stage 3. Training Q-function. For the convenience of observation variables that accumulate data along the length of the trajectory and the total reward for episodes are initialized. Next, Q- function is created in the form of the table (shown in Tab. 2) and it is filled with random numbers.

Table 2. The table Q-function

action a_n / **state** S_n	**North**	**South**	**East**	**West**
S_0	0.010212828	0.010247146	0.012362903	0.010169895
S_1	0.010286808	0.012487781	0.010350737	0.010269896
S_2	0.010315242	0.010452861	0.01045163	0.010368139
. . .				
S_{9999}	0.017854595	0.021050301	0.022042031	0.048342432

During simulation a deterministic strategy is used, that is, the choice of actions is carried out strictly according to the criteria of Q-function. Therefore, it is important to assign values in the table so that the agent can reach the goal later. The table can not be initialized with zeros. Each new episode starts from the beginning. One step of training is as follows:
Step 1. Select an action $a^{'}$ by strategy using the current Q-function.
Step 2. Action message $a^{'}$ to simulation environment, obtaining a new state $S^{'}$ that depends on the selected step, the reward r and signal of the end of the episode.
Step 3. Calculate the new target value Q_{aim} for Q-function according to (4):

$$Q_{aim} = \begin{cases} r + \gamma \max_{a'} Q(S^{'}, a^{'}) \\ r \quad \text{if done} \end{cases} . \tag{5}$$

If simulation episode is over then the target value is $Q_{aim} = r$. Otherwise, the value Q_{aim} will be updated on the bases of the discount rate γ set in stage 2 (according to the Bellman equation).
Step 4. Taking into account the learning rate α set at the stage of forming training parameters, update scales of Q-function for pairs of states and actions $Q(S, a)$ at the end of each episode. Updating values of Q-function can be represented by the following expression

$$Q(S, a) \leftarrow (1 - \alpha)Q(S, a) + \alpha Q_{aim}. \tag{6}$$

The process of learning how to build an optimal UAV route using the table Q-learning algorithm is represented by linear plots in Figs. 2 and 3.

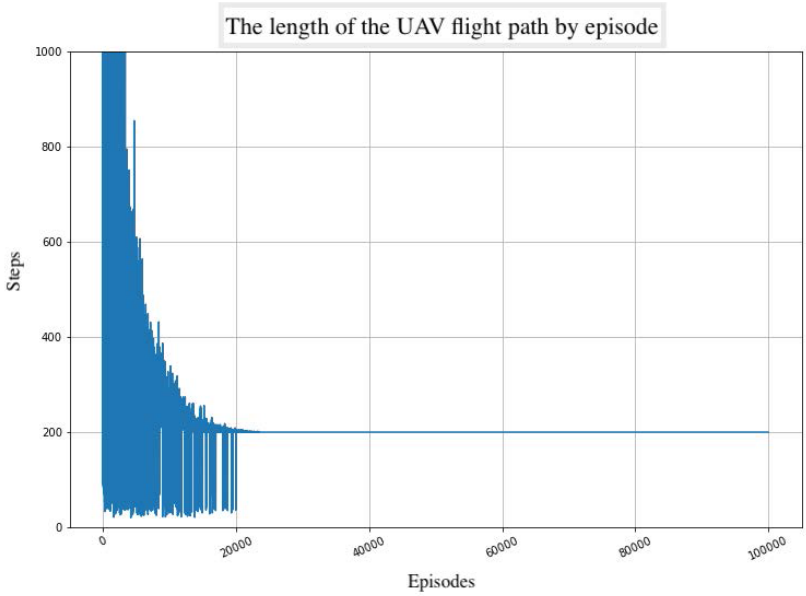

Fig. 2. The length of the UAV flight path by training episodes

The trajectory length history plot shows that the UAV performed many actions during the first 20,000 episodes. But later, due to the update of scales of Q-function it learned to reach the goal in 200 steps.

Similar relationship is observed in the graph of total rewards by episode. The agent's reward in the early stages of the simulation is equal to 0 because it either did not reach the goal or it

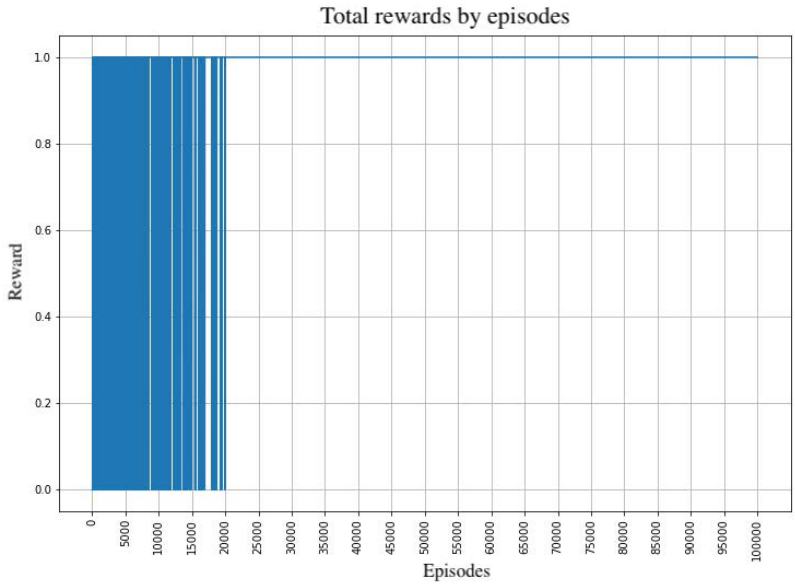

Fig. 3. The total rewards by training episodes

could not complete the episode in the maximum number of steps. However, in some episodes it still managed to get the maximum reward. After 20,000 episodes weights of Q-function reached the optimal level. It results in the stable assignment of the maximum reward, that is, the UAV began to reach the goal. Step 4. Start the test simulation. The optimal strategy on the trained Q-function is used. Three hundred episodes are generated to test the performance of the algorithm. The probability of successful route construction is 88 % . A variant of the generated UAV path is shown in Fig. 4, where the red zone is the radius of damage by the air defence system (AD).

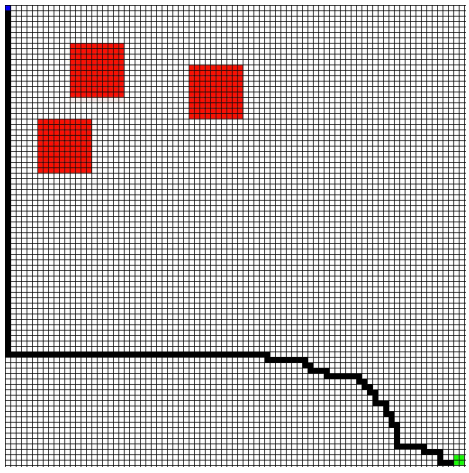

Fig. 4. An example of the UAV flight route in the conditions of counteraction calculated according to the algorithm of the table Q-learning

In comparison with the traditional algorithm for constructing a UAV route using a random direction vector generator the table Q-learning is more intelligent. The UAV motion vector is obtained according to the policy, and random number generator is not used. However, with this approach the agent may find trajectory that is not the most efficient trajectory, and because of the deterministic strategy the agent follows it.

2.4. The ε-greedy strategy of Q-learning for constructing the optimal UAV route

How to prevent situations when agent that follows the optimal policy is trapped in a local maximum and receive a guaranteed but low reward. Instead of immediately making a move based on deterministic strategy, one can consider other areas of the map in search of the best optimal route. However, excessive search may result in the lack of reward. Insufficient search of the environment may result in the local maximum trap. Therefore, a balance between the use of optimal policy and environmental search is needed.

The strategy for balancing between the use of optimal policy and environmental search in the RL is called ε– greedy strategy. Let consider some fixed parameter ε that takes values from 0 to 1. It is the probability that UAV will choose a random action, and $1-\varepsilon$ is the probability that it will be guided from the current optimal policy just like with the classic Q-learning. This strategy can be represented as follows

$$S \to \begin{cases} random\ a \ \text{ if } \ P = \varepsilon \\ \arg\max\ Q(S,a) \ \text{ if } \ P = 1-\varepsilon \end{cases} \to a^{'} \to S^{'}. \tag{7}$$

During training process, parameter ε should be changed. In the early stages, until weights of Q-function are not optimized enough, one needs to set high values of ε for the agent to explore the environment. Towards the end of training when environment is sufficiently studied and one should rely on the optimal strategy one needs to reduce parameter ε to zero. Obviously, ε is decreased linearly with the episode number.

Stages of building an optimal UAV route in conditions of active obstacles by ε - greedy policy do not differ from the table Q-learning with the exception of selecting an action in accordance with expression (7). Hyperparameters of training are similar to parameters of Q-learning. To maintain the balance between two strategies parameter $\varepsilon = 0.9$.

The learning process is also represented by graphs of the history of trajectory lengths and total rewards for training episodes. They are shown in Figs. 5 and 6 below.

The plot of the UAV flight path lengths shows that when in the early episodes of the greedy strategy the agent explores environment it enters the air defence zone and the allowed number of steps per training iteration is exceeded. Later, when parameter ε is reduced linearly on the bases of updated weights of Q-function the agent begins to reach the goal in a fixed number of steps. However, in comparison with the tabular Q-learning the intelligent agent studies the area in detail (more precisely, function weights $Q(S,a)$), and there is the variety of the choice of path.

The plot of total rewards for ε-greedy policy shows chaotic assignment of total rewards especially this trend is in evidence in the middle of training. This is because the agent has to choose a random action to explore the area rather than to follow a strict selection policy.

Let us perform a quality check for updating the function weights $Q(S,a)$ similar to checking the table Q-training, and take into account the exploration strategy. The probability of successful construction of UAV routes here is 92%. The increase in the probability of successful construction

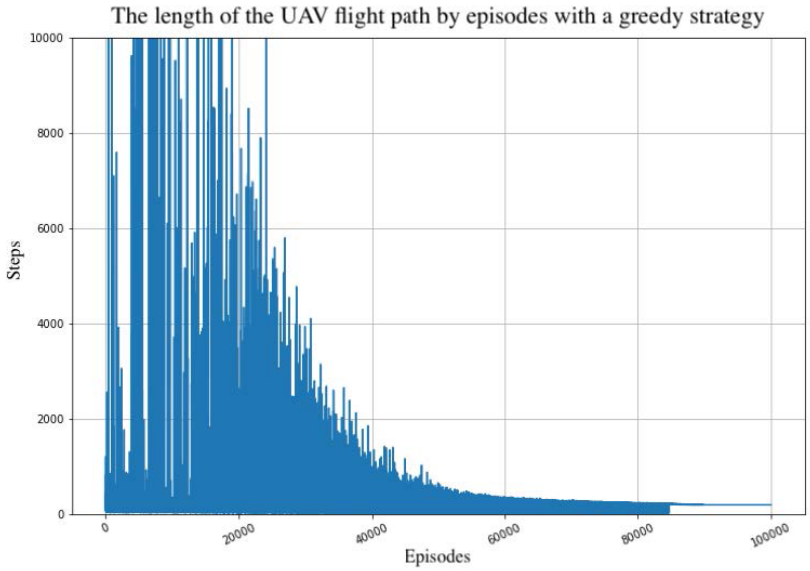

Fig. 5. The lengths of the UAV flight paths with the ε-greedy strategy

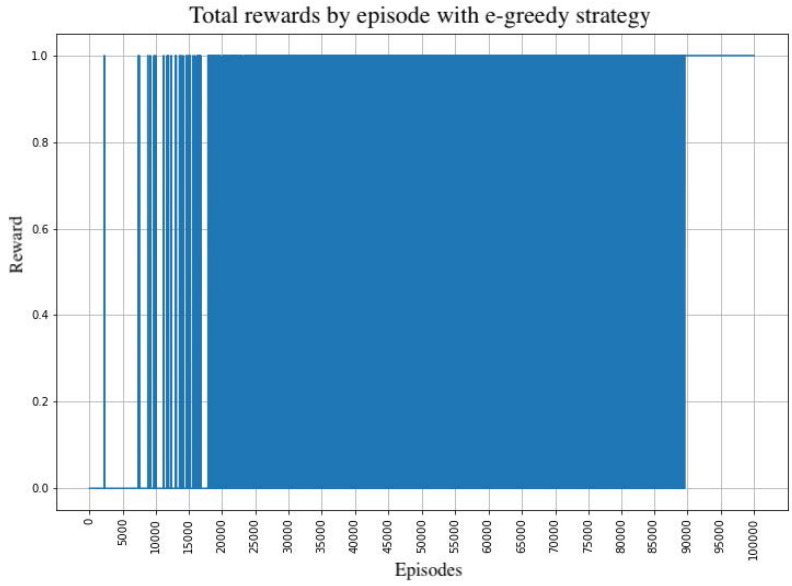

Fig. 6. The total awards by training episodes with the ε-greedy strategy

is explained by a more detailed update of Q-function at the expense of ε-greedy strategy. Figure 7 shows the UAV path constructed using ε-greedy approach of RL.

For more complex and real tasks Q-function can not be represented as a table because the size of the table would require huge amount of memory. Therefore, one needs to approximate

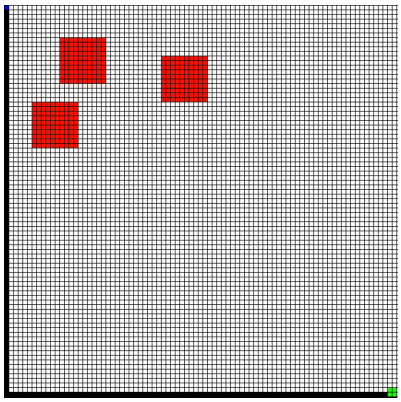

Fig. 7. The generated UAV flight route under counteraction conditions using ε-greedy Q-learning method

the function of evaluating pairs of states and actions.

3. Construction of the optimal UAV route in the conditions of counteraction with the use of deep Q-learning

Deep Q-learning algorithm (DQN) uses a neural network for approximation of Q-functions. Now instead of updating the weights of function $Q(S, a)$ the state of the environment S is transmitted directly to the neural network, and the network will return the values of Q for each possible action [8].

Training stages of DQN are similar to that of the Q-learning algorithm. However, the approximation requires some modernization. One must introduce a parametric function Q_W which predicts the quality of the pair $Q(S, a)$ and it depends on the parameters of the approximator W.

Approximation of Q-function can be formulated as a supervised learning. Let Q_{aim} be the target value based on expression (4). In DQN it is presented as follows

$$Q_{aim} = r + \gamma \max_{a'} Q_W(S', a'). \tag{8}$$

One needs to update settings W so that function Q_W is to be close to the target values Q_{aim}. In supervised learning, one needs to formulate the loss function E which is calculated in DQN as follows

$$E = (Q_{aim} - Q_W(S, a))^2. \tag{9}$$

Using the method of gradient descent, error is minimized by updating the weights of the approximator:

$$W \leftarrow W - \alpha \frac{\partial E}{\partial W}, \tag{10}$$

where α is learning rate.

A neural network model is used to approximate the optimal Q-function. TensorFlow and Keras libraries for machine learning were used to build the network architecture. Formally it is a neural network based on TensorFlow with one Embedding layer that converts positive indices to dense vectors of fixed size [9]. States S are transformed into a vector $\overrightarrow{Q(S)}$ for various actions.

The square of the norm of the difference between the target and predicted vector of Q-function is used as a loss function as shown in (9). The stochastic gradient descent optimizer is used as an optimizer for the neural network. DQN training is similar to the table Q-learning. To select the direction vector the ε-greedy approach is used. Since the neural network predicts vector $\overrightarrow{Q(S)}$ for all possible actions it is necessary to determine how to obtain the goal vector Q_{aim}.

Suppose that target vector $\overrightarrow{Q(S)}$ is equal to basic values for all actions except for the action a_i. The task of the algorithm is to train the neural network on this target vector in such a way as to change it's weights so that only the value for a_i is updated and they didn't change for other actions. This strategy is implemented by minimizing the error.

According to the algorithm, one minimization step requires one step of gradient descent through the built in batch gradient descent function (batch-size) [9, 10]. It implements the ability to evaluate subsets of a fixed-size training sample [11]. In the considered algorithm, the input is a subset of the training sample with a length of one element. The neural network training developed by the algorithm is identical to the traditional approaches of the RL. However, for the weights of the goal vector $\overrightarrow{Q(S)}$ to be updated with each training step $\alpha = 0.1$ is assigned.

The training progress of the neural network is shown in Figs. 8 and 9.

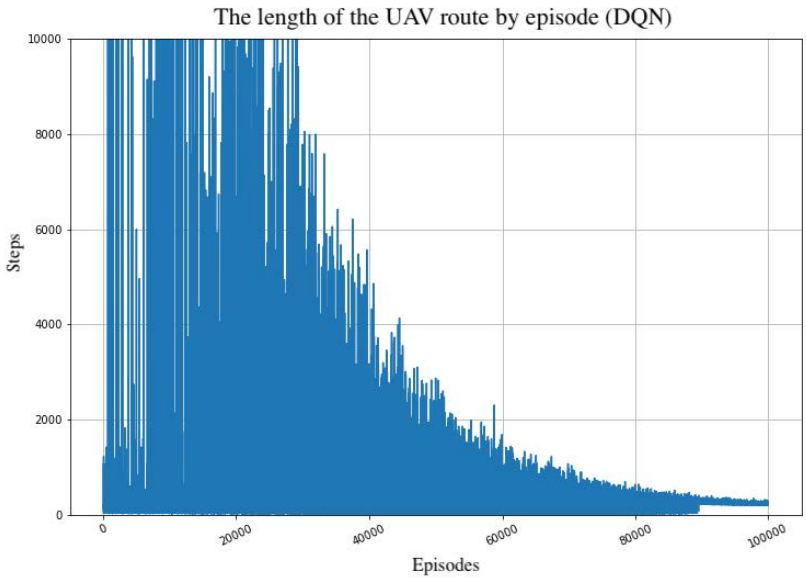

Fig. 8. The length of the UAV flight paths by episodes in the process of training of the DQN neural network

The plot shows an identical behaviour of the agent. It is the same as in the case of ε-greedy strategy. This is because the greedy policy was chosen to select the direction vector of the agent.

The graph differs from the one presented earlier. On the abscissa not the number of episodes is shown but the reference points where the total rewards were averaged for every 100 training episodes.

To evaluate the quality of the trained neural network model 300 verification episodes similar to the classic RL approaches were generated. The probability of successfully calculated UAV

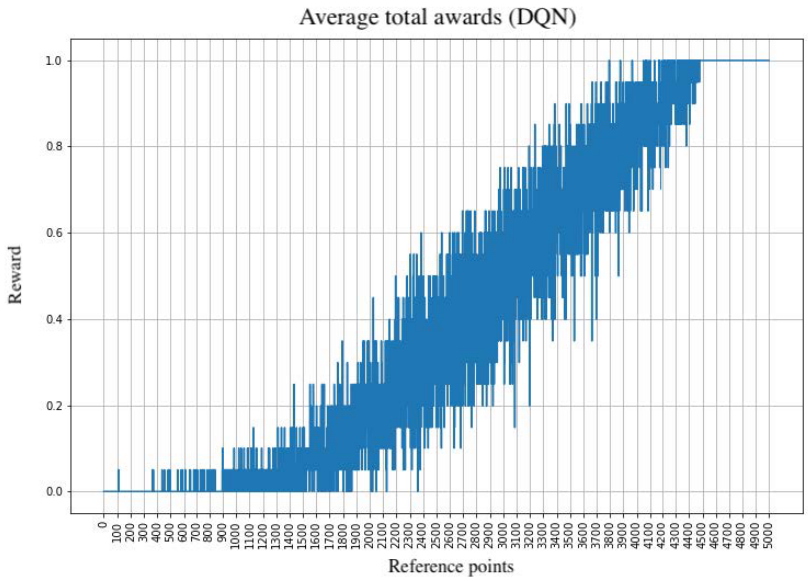

Fig. 9. The total averaged rewards of the agent under the DQN policy of choosing direction vector

routes is 97%. Fig. 10 shows an example of the optimal UAV route constructed using neural network algorithm DQN.

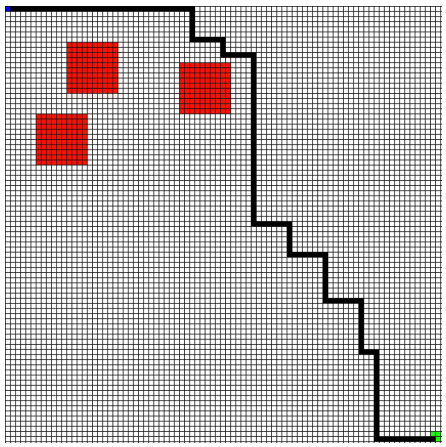

Fig. 10. The optimal UAV path designed with deep Q-learning

4. Comparative characteristics of algorithms

The interactive cloud environment Google Colab was used for training and performing calculations. It has a GPU NVIDIA TESLA T4 with 16 GB of memory. The effectiveness of presented algorithms was evaluated in terms of the following parameters [12]: training time, the average length of the UAV flight route trajectory, the probability of successful route construction, and

the minimum number of episodes before receiving the maximum reward. The simulation results are presented in Tab. 3.

Table 3.

Algorithm	Training time (hr:min:sec)	Min number of episodes to receive Max award	Average path length, step	Probability of successful construction routes
Traditional algorithm	Less than 1 min.	–	531	67
Table Q-learning	6 min. 46 sec.	20000	200	88
ε-greedy policy	16 min. 37 sec.	83000	205	92
DQN	22 hr. 14 min.	88000	250	97

The traditional algorithm requires the least training time. However, the probability of successful route construction is much lower in comparison with OP algorithms. DQN requires more time and considerable computing resources. For comparison, training without the use of a GPU takes more than 2 days. However, the probability of successfully generated trajectories is close to 100%.

Conclusion

The results of simulation prove that developed algorithms can be used to solve the problem of constructing the optimal flight route of an unmanned aerial vehicle in the conditions of counteraction. Each algorithm has its own advantages and disadvantages. Nevertheless, artificial neural networks significantly outperform traditional algorithms in terms of the quality of obtained results. They can be used to solve more complex problems of the optimal route construction, including the case of dynamically changing environment.

The work was partially supported by the Russian Science Foundation grant no. 22-21-00001.

References

[1] A.P.Tanchenko, A.M.Fedulin, R.R.Bikmaev, R.N.Sadekov, UAV Navigation System Autonomous Correction Algorithm Based on Road and River Network Recognition, *Journal Gyroscopy and Navigation*, (2020), no. 11, 293–299 (in Russian).

[2] I.N.Ishchuk, A.A.Dolgov, Computer model and algorithm of construction of phono-target situation of remote monitoring areas by thermal tomograms taking into account their geographical location and meteorological conditions, *J. Sib. Fed. Univ. Eng. & Technol.*, **13**(2020), no. 3, 350–360 (in Russian).

[3] I.N.Ischuk, A.A.Dolgov, M.A.Likhachev, B.K.Telnykh, Model for calculating the thermophysical characteristics of materials according to multispectral multi-temporal photographic survey of the earth's surface, *Journal of Siberian Federal University, Technics and technologies*, **13**(2020), no. 7, 906–918 (in Russian).

[4] R.S.Sutton, A.G.Barto, Reinforcement Learning: An Introduction, A Bradford Book Publ, 1998.

[5] B.Nikhil, L.Nicholas, Fundamentals of Deep Learning. Designing Next-Generation Machine Intelligence Algorithms, Moscow, Mann, Ivanov and Ferber, 2020 (in Russian).

[6] Yu.Liu, PyTorch 1.x Reinforcement Learning Cookbook, Moscow, DMK Press, 2020 (in Russian).

[7] Ch.Yan, X.Xiang, Ch Wang, Towards Real-Time Path Planning through Deep Reinforcement Learning for a UAV in Dynamic Environments, *Journal of Intelligent & Robotic Systems*, 2019.

[8] V.Mnih, Human-level control through deep reinforcement learning, *Nature*, **518**(2015), no. 518, 529–533.

[9] Keras API docs, *official site: https://keras.io/api*, 2021.

[10] I.Goodfellow, Y.Bengio, A.Courville, Deep Learning, MIT Press, 2016.

[11] D.R.Liu, H.L. i, D.Wang, Feature selection and feature learning for high-dimensional batch reinforcement learning: A survey, *Int. J. of Automation and Computing*, **12**(2015), 229–242.

[12] D.A.Korobov, S.A.Belyaev, Modern approaches to training intelligent agents in the atari environment, *Software & Systems*, **31**(2018), no.2, 284–290 (in Russian).

Интеллектуальные подходы к решению задачи построения оптимального маршрута беспилотного летательного аппарата в условиях противодействия

Игорь Н. Ищук
Сибирский федеральный университет
Красноярск, Российская Федерация
Богдан К. Тельных
Военный учебно-научный центр Военно-воздушных сил
«Военно-воздушная академия имени профессора Н.Е. Жуковского и Ю.А. Гагарина»
Воронеж, Российская Федерация
Валерий Н. Тяпкин, Николай С. Кремез
Сибирский федеральный университет
Красноярск, Российская Федерация

Аннотация. В статье представлены традиционные и современные алгоритмы к решению задачи планирования оптимального маршрута беспилотного летательного аппарата в условиях воздействия на него маловысотных систем противовоздушной обороны. Описаны принцип работы предложенных способов, а также использующиеся в них инструменты. Рассмотрены классические подходы обучения с подкреплением и его модификация с использованием искусственных нейронных сетей.

Ключевые слова: беспилотный летательный аппарат, искусственные нейронные сети, информационные технологии, интеллектуальные агенты, обучение с подкреплением.