The Method of Intelligent Railway Alignment Path Generation Based on Deep Q Network

Tianxi Wang*, Baocheng Wang

School of Civil Engineering, Lanzhou Jiaotong University, Lanzhou 730070, China

Corresponding author's e-mail: 804632281@qq.com

Abstract. Traditional railway route selection requires manual fieldwork over long periods, which is physically demanding and subject to seasonal weather conditions, leading to an uneven annual production cycle and low efficiency. With the introduction of high-tech methods and the significant development of artificial intelligence, AI has become practical. Deep reinforcement learning, with its perceptual and decision-making capabilities, is well-suited to address route planning problems. It can be applied to modern route selection techniques by training the exploration ability of the established model using the DQN algorithm. The intelligent agent receives positive rewards when approaching the target point and negative rewards when moving away from it, aiming to optimize construction costs. Experimental results demonstrate that compared to manual selection, the intelligent route selection approach yields similar paths while significantly reducing labor costs and saving approximately 11.3% in construction expenses.

Keywords: Deep reinforcement learning:Intelligent line selection;Optimal path;DQN algorithm

1 INTRODUCTION

Today, the development of transportation infrastructure is rapid. Railways, as the backbone of the comprehensive transportation system, possess advantages such as high transport capacity, fast speed, long-distance transport superiority, stability and reliability, lower energy consumption, and environmental friendliness.[1] Railway alignment, known as the "leader" of railway engineering construction, is the foundation of railway construction and a comprehensive task involving various factors. It determines the direction of new railway construction or railway renovation and provides a basis for subsequent survey and design. Alignment considerations include political, economic, technical, and geographical factors to achieve objectives of technological advancement, economic feasibility, safety, and convenience of operation.[2-3] Traditional railway alignment requires railway designers to conduct on-site surveys and measurements, selecting routes through repeated comparisons. This method is direct, simple, and allows for clear understanding of terrain and features along the railway alignment, resulting in reliable route selection.[4] However, drawbacks include the need for designers to work extensively in the field, leading to high workload, physical strain, and exposure to outdoor conditions, including seasonal and climatic influences. This method also poses high risks in the field, presenting challenges to designers.

With the development of the times, the introduction of digital technology, modern surveying technology, artificial intelligence, and other modern technologies into railway alignment design will bring significant changes to traditional alignment methods. Through the updating and improvement of modern alignment technologies, multiple reasonable route selection schemes can be automatically generated by computers or supercomputers, providing references for designers. This can reduce the workload of designers, improve work efficiency and quality, and effectively avoid overlooking valuable alignment schemes. Route optimization based on genetic algorithms, ant colony algorithms, and *A** algorithms has become a research hotspot in recent years. In 2006, Hinton from Canada and his student Salakhutdinov first proposed the concept of deep networks. Pu H, Zhang H, et al. designed a convolutional neural network model for making decisions on the maximum gradient of railways. This model takes images cropped from the complete map as input data and outputs the maximum gradient after processing through the convolutional neural network. By leveraging deep learning in a data-driven manner, decisions on the maximum gradient for different terrains and railway grades are made. The DeepMind team employed a small number of expert samples to achieve human-like motion control for robots through imitation learning. Finn et al. combined deep reinforcement learning with prediction of robot grasping actions, enabling self-supervised training for image prediction while training robots to execute strategies. In addition, deep reinforcement learning has been applied in various fields such as autonomous driving, natural language processing, control optimization, object localization, machine translation, text generation, video prediction, and more. Despite its application in various industries, deep reinforcement learning has not yet been fully developed for the automatic generation of railway routes. The preliminary exploration of deep reinforcement learning in the field of railway route selection design is of great significance for advancing the development and application of artificial intelligence in route selection, emphasizing its importance.

2 DEEP REINFORCEMENT LEARNING

2.1 The Principle of Reinforcement Learning

The principle model of reinforcement learning is illustrated in Figure 1. Suppose there is an individual, referred to as the intelligent agent, capable of executing action policies within the environment. At time t, the intelligent agent perceives the current state information of the environment, denoted as St, and outputs an action at through its policy. The intelligent agent executes action at, which affects the environment and transitions to a new state, S_{t+1}, while also receiving a value reward,$r(s_t,a_t)$, from the environment. This process continues in a loop, where the intelligent agent accumulates experience and adjusts its policy through continuous interaction with the environment. Ultimately, the intelligent agent aims to derive the optimal action policy to maximize the cumulative value rewards when completing tasks. [5-6]

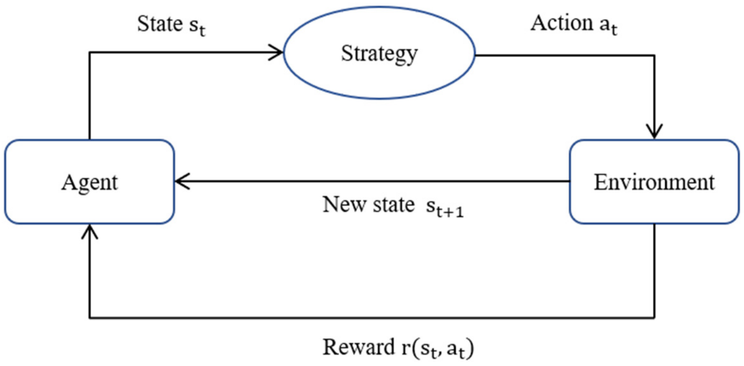

Fig. 1. Theory framework for reinforcement learning.

The Q-learning algorithm (QL) is a classical reinforcement learning algorithm used to solve decision problems in Markov Decision Processes (MDPs). It guides the agent in selecting the optimal actions within the environment by learning a value function known as the Q-value function. The fundamental idea of the Q-learning algorithm is to continuously update the Q-values to learn the optimal policy, enabling the agent to make optimal decisions within the environment. [7-8] The Q-values are updated using the Bellman equation to approximate the true optimal Q-values:

$$Q(s,a) \leftarrow Q(s,a) + \alpha\left(r + \gamma \max_{a'} Q\left(s', a'\right) - Q(s,a)\right) \tag{1}$$

In this context: s is the current state, a is the action chosen in the current state, r is the reward obtained after taking the action, s' is the next state after taking the action, α is the learning rate (which controls the step size of each update), and γ is the discount factor (which balances the importance of current and future rewards). Both α and γ are within the range *[0,1]*. Once each state-action pair in the Q-value table has been visited and updated a sufficient number of times, the Q-table converges, and the agent will receive the maximum cumulative reward when completing tasks. This leads to the optimal policy, enabling the agent to make optimal decisions within the environment.[9]

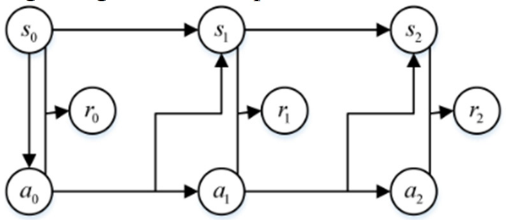

Fig. 2. Markov decision-making process diagram.

Railway intelligent alignment and exploration reinforcement learning share the same theoretical foundation, as they both adhere to the framework of Markov Decision Processes (MDP). Markov Decision Processes (MDP) are a mathematical framework, illustrated in Figure 2, used to model sequential decision problems with both randomness and decision-making. It is capable of detecting ideal states, allowing for multiple attempts, where the next state of the system only depends on the current information state and is not influenced by earlier states. The decision-making process is also associated with the current action to be taken. [10]

In practical applications, there are limitations to iteratively solving the optimal policy using this method. When the environment dimensions are high and there are many states, it can impose a significant computational burden on the computer, known as the "curse of dimensionality".

2.2 The Deep Reinforcement Learning Algorithm

To address the curse of dimensionality, some scholars have proposed Deep Reinforcement Learning, which combines the excellent perceptual abilities of deep learning with the decision-making capabilities of reinforcement learning. It utilizes deep neural networks $Q(s,a;\theta)$ to approximate the Q-table function, where θ represents the parameters of the neural network. The neural network takes a state vector as input and outputs the value function for each action. [11] This approach is known as the Deep Q Network (DQN) algorithm. Initially, two structurally identical neural networks are created: one is the value function network (also known as the target network), and the other is the network used to approximate the value function (also known as the evaluation network). Then, the difference between these two networks is computed. Subsequently, the current action-value function is updated using the temporal difference method. Therefore, the loss function for each iteration of Deep Q Learning can be described as:

$$L_i(\theta_i) = E_{s,a,r,s^i}\left[\left(y_i^{DQN} - Q(s,a;\theta_i)\right)^2\right] \tag{2}$$

In the above formula, y_i^{DQN} represents the temporal difference target.

$$y_i^{DQN} = r + \gamma \max_{a'} Q(s',a';\theta^-) \tag{3}$$

Through experience replay, the temporal difference algorithm can enhance the effectiveness of neural networks. In this process, the agent accumulates experiences $t_i=(s_i,a_i,r_i,s_{i+1})$ from multiple interactions and stores them in the experience pool D, forming a buffer space, where $D=t_1,t_2,...,t_i$ represents different time steps. Then, the agent randomly samples from this experience pool to use samples for training the neural network. Thus, we can express its loss function as follows:

$$L_i(\theta_i) = E_{s,a,r,s'u(D)}[(y_i^{DQN} - Q(s,a;\theta_i)^2] \tag{4}$$

The purpose of experience replay is to randomly sample from stored experiences, thereby breaking the correlation between data, which helps improve the stability and convergence of neural networks.

3 THE RAILWAY INTELLIGENT ALIGNMENT MODEL BASED ON GRID TERRAIN

3.1 The Model Environment

The environment design area is a vast, continuous three-dimensional space. If the alignment model's environment consists of continuous states, the agent's trajectory has countless possibilities, making it impossible to complete the route search command. Therefore, we assume that the environment for finding the optimal route alignment is defined as a collection of states composed of grid cells. The actions of railway route selection involve sequential decision-making (sequential decision-making refers to decisions arranged in chronological order to obtain various decisions in sequence), complying with the delayed feedback rule in reinforcement learning. Reinforcement learning can effectively capture real-time changes in states, making it suitable for solving the problem. We use the Markov Decision Process for modeling. By using the Digital Elevation Model (DEM) from GIS systems, we discretize the continuous space into a three-dimensional digital grid map with elevation information. Points such as the starting point, endpoint, terrain, landforms, hydrology, areas of geological instability, and environmentally sensitive areas in the alignment geographical environment are marked as "C_i", serving as the model environment for railway intelligent alignment. [11]

$$E=\{E_1,E_2,\cdots,E_n \mid E_i=(X_i,Y_i,Z_i,C_i)\} \tag{5}$$

The intelligent agent's planar coordinates are denoted as $L_i=(X_i,Y_i)$, the elevation information at the planar coordinates is denoted as Z_i, and terrain attributes are denoted as C_i. The starting point is labeled as "Start," and the endpoint is labeled as "Goal." Areas designated as environmentally sensitive or geologically unstable are marked as forbidden zones Forbidden, while other areas are considered Normal.

3.2 Model Traversal States

For the problem of route planning, it is necessary to simplify the vehicle to a point mass for simulation. That is, setting an agent to execute pathfinding commands in the simulated terrain, where the position of the agent can be considered as a state within a grid cell. However, the function of intelligent route selection is to find a standard line type that meets the specifications. Therefore, the agent should avoid situations where the curve radius does not meet the requirements during the selection of the route path, such as when the curve radius is too small, causing the train, which is not a point mass but has a length, to be unable to pass through sharp turns. The transition from the previous position Li-1 to the current position L_i and then to the next position L_{i+1} may result in significantly different turning angles, as shown in Figure 3. [12] Therefore, the train cannot be simply converted into a point mass, and the length of the train itself must be taken into account.

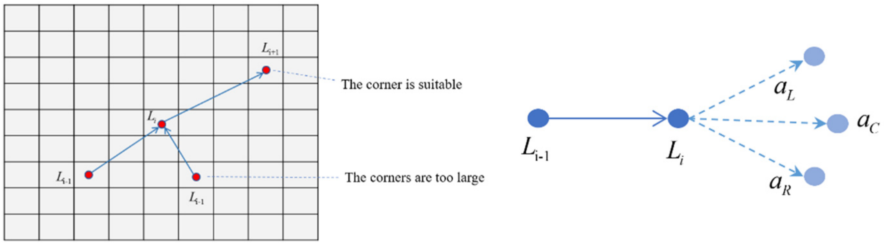

Fig. 3. Different angles caused by single location and Create three-way action

In this paper, the model connects two adjacent locations to form linear units, which serve as the position states of the agent. The direction of the route is determined by the angle between two linear units. The agent starts from the starting point and reaches the target point, obtaining an initial line shape fitted by many linear units. The collection of all linear units in the environmental space is represented as the state space *S*.

$$S = \{S_1, S_2, \cdots, S_n \mid S_i = (X_{i-1}, Y_{i-1}, X_i, Y_i)\} \tag{6}$$

For the railway alignment task, it is recommended not to set the exploration step size L_i for the agent too small. A step size that is too small can lead to two issues: first, it significantly reduces the convergence speed of the model; second, if the step size is too small, the distance between adjacent positions is too small, which may cause the route to visually appear with spikes and excessive turning angles that do not match reality. Therefore, it is suggested to use a DEM model with a grid precision of 30 meters, allowing the agent to cross N elevation points per step, where N is a variable hyperparameter. Although there are eight directions available for each position (east, west, south, north, northeast, northwest, southeast, southwest), to ensure smooth turning of the route, the action space A_i is restricted to three directions relative to the current heading: left forward, straight forward, and right forward. The update of the agent's state is as follows, where a_{ix} and a_{iy} represent the components of action a_i in two directions.

$$\begin{gathered} A_i = \{a_L, a_C, a_R\}, a_i \in A_i \\ L_{i+1} = (X_i + a_{ix}, Y_i + a_{iy}) \\ S_{i+1} = (X_i, Y_i, X_i + a_{ix}, Y_i + a_{iy}) \\ l_i = \sqrt{a_{ix}^2 + a_{iy}^2} \end{gathered} \tag{7}$$

3.3 The Model's Reward Function

The reward function determines the objective characteristics that the agent needs to optimize when employing expert strategies. When the agent perceives the environmental state, the reward function provides a reinforcement signal r to evaluate the impact of its actions on the environment. Therefore, we can establish a set of evaluation rules based on changes in environmental attributes, denoted as Equation 8.

$$\text{Cost}_i = C_{T_i} + C_{S_i} + C_{B_i} + C_{L_i}$$

$$\text{Reach}_{i+1} = \begin{cases} \text{Bonus}, C_{i+1} = \text{Goal} \\ 0, C_{i+1} = \text{Normal} \\ -\text{Penalty}, C_{i+1} = \text{Forbidden} \end{cases} \tag{8}$$

$$R_i = -\text{Cost}_i + \text{Reach}_{i+1}$$

In the equation: C_{Ti} represents the cost of tunnel engineering; C_{Si} represents the cost of embankment engineering; C_{Bi} represents the cost of bridge engineering; $\mathrm{C_{Li}}$ represents the cost related to the length of the railway, communication, and power aspects; C_{osti} represents the engineering cost involved in transitioning from state Si to state S_{i+1}; $Reach_{i+1}$ represents the feedback values given based on the attributes of the new state S_{i+1}. When the agent reaches the destination point, it will receive an additional positive reward, Bonus; when entering a forbidden area, it will receive a penalty, Penalty, and then start the next round of exploration. In other connectable sections, the agent will continue its exploration for this round and will not receive additional feedback signals.

3.4 $\varepsilon - greedy$ Policy.

The $\varepsilon - greedy$ policy is a commonly used exploration strategy in reinforcement learning, employing the $\varepsilon - greedy$ strategy as the agent's action selection policy. In this strategy, ε represents the greediness level, with values ranging from 0 to 1. When executing actions, the agent randomly selects from feasible actions with a probability of ε, while with a probability of 1-ε, it chooses the action with the highest estimated value. In the initial exploration phase, due to the agent's limited environmental experience, ε is set relatively high to encourage more random exploration and avoid local optima. As the number of explorations increases, the value of ε gradually decreases, causing the agent to prefer actions with the highest estimated value to effectively utilize accumulated environmental experience.

should be numbered with a dot following the number and then separated by a single space:

$$\pi(a \mid s) \leftarrow \begin{cases} 1 - \varepsilon + \dfrac{\varepsilon}{|A(s)|}, \\ \text{If } a = \arg\max_a Q(s,a); \\ \dfrac{\varepsilon}{|A(s)|}, \text{ If } a \neq \arg\max_a Q(s,a) \end{cases} \tag{9}$$

4 THE SIMULATED EXPERIMENTAL RESULTS

Based on the terrain of a mountainous area along the Hubei Yangtze River Railway, reasonable learning rates and discount rates were set to prevent the agent from prematurely converging to local optima and to balance between short-term and long-term benefits. Specifically, in this experiment, the learning rate was set to 0.01, the discount rate was set to 0.9, the epsilon value for epsilon-greedy strategy was set to 0.1, the maximum number of episodes was 300, the capacity of the experience replay buffer was set to 500, and every 500 time steps, the weights of the evaluation network were copied to the target network. The maximum number of iterations per episode was set to 10000. The experimental process was visualized, showing the interaction between the agent and the environment graphically, and the paths were plotted in the simulated environment, as shown in the Figure 4.

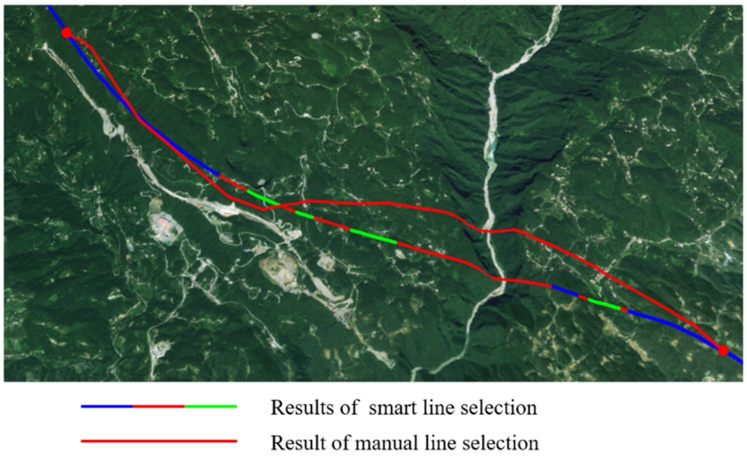
Results of smart line selection
Result of manual line selection

Fig. 4. Comparison of intelligent railway location design.

After conducting exploration in the environment for one hundred thousand rounds, the agent obtained 26 alternative route plans. To further evaluate its performance, we selected one representative route plan as the optimal intelligent route selection solution and compared it with the manually designed route plan. Table 1 shows the comparison of model parameters and engineering costs.

Table 1. Model parameters and Engineering cost comparison

Project	Numerical value	Project	Manual scenarios	Intelligent line selection
Study area/km^2	46.54	Total length of the line/km	9.520	9.236
Map accuracy/m	30	Tunnel length/km	2.842	3.228
Action step size N	5	The length of the roadbed/km	2.583	1.534
Learning rate α	0.01	H<40m Bridge	2.245km	2.524km

Discount factor γ	0.9	H>40m Bridge	1.850km	1.950km
Greed ε	0.1	Total cost/$10^4 CNY$	16 934.8	15 017.6
Memory pool capacity	500	Cost savings/%	—	11.3

The intelligent route selection scheme and the manual scheme exhibit overall similarities in their trajectories, but significant differences arise in crossing rivers. The manual approach involves crossing valleys directly with small-radius curves, leading to the need for numerous bridges and tunnels, thus resulting in higher construction costs. In contrast, the intelligent route selection scheme adopts large-radius curves to meander around, balancing the length of the roadbed and bridges by finding optimal crossing points, thereby saving a significant amount of construction costs.

5 CONCLUSION

- The Deep Reinforcement Learning algorithm is an improvement upon the traditional Q-learning algorithm, aimed at addressing the limitation of the curse of dimensionality in Q-learning. By utilizing neural networks to approximate the state-action value function, Deep Q Network effectively avoids the problem of dimensionality curse inherent in tabular Q-learning;
- The experiments show that the intelligent route selection scheme has a better route selection than manual route selection, not only greatly freeing up manual costs, but also saving about 11.3% of construction costs;
- The route selection design involves the comprehensive consideration of various objectives such as engineering construction, operation and maintenance, safety and comfort performance, and regional transportation needs. However, the optimization objectives of this study have not fully covered all aspects, so further improvements will be made in future research. Reinforcement learning has great potential in railway route selection and holds promising prospects for development.
- This paper only adopts one form of the DQN algorithm, which predicts and approximates the value function by identifying the agent's chosen actions. However, the deeper essence of the DQN algorithm lies in using image recognition to learn the optimal strategy. It utilizes convolutional neural networks to transform images from high-dimensional perceptual recognition to low-dimensional abstract feature representations, and learns strategies from them. Therefore, the author's next step will be to explore the deeper utilization of the DQN algorithm.

REFERENCES

1. Li, W., Pu, H., Zhao, H., & Liu, W. (2013). Approach for Optimizing 3D Highway Alignments Based on Two-stage Dynamic Programming. *J. Softw.,* 8, 2967-2973.
2. Feng, S., Shu., H, & Xie, B. (2021). 3D environment path planning based on improved deep reinforcement learning[J]. Computer Applications and Software, 38(1): 250−255.
3. Li Q, Lin T, Yu Q, Du H, Li J, Fu X. (2023). Review of Deep Reinforcement Learning and Its Application in Modern Renewable Power System Control. Energies. 16(10):4143.

4. Liu Q, Zhai J, Zhang Z, et al. (2018). A survey on deep reinforcement learning[J]. Chinese Journal of Computers, 2018, 41(1): 1−27.
5. Guo X, Fang Y. (2018). Introduction to the principles of reinforcement learning. [M].Electronic Industry Press, Beijing.
6. Choi, J., Lee, G. & Lee, C. (2021).Reinforcement learning-based dynamic obstacle avoidance and integration of path planning. *Intel Serv Robotics* 14, 663–677.
7. Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., & Riedmiller, M.A. (2013). Playing Atari with Deep Reinforcement Learning. *ArXiv, abs/1312.5602*.
8. Ye H, Tu W, Ye H, et al. (2020). Deep reinforcement learning based electric taxi service optimization[J]. Acta Geodaetica et Cartographica Sinica, 49(12): 1630−1639.
9. Mondal S , Lucet Y , Hare W . (2015). Optimizing horizontal alignment of roads in a specified corridor[J].Computers & Operations Research, 64(C):130-138.
10. De Smith M J. (2006). Determination of Gradient and Curvature Constrained Optimal Paths[J]. Computer-Aided Civil and Infrastructure Engineering, 21: 24-38.
11. Zhou C, Gu S, Wen Y, et al. (2020). The review unmanned surface vehicle path planning:based on multi-modality constraint[J]. Ocean Engineering, 200: 107043.
12. Xue, X., Li, W., & Pu, H. . (2018). Review on intelligent optimization methods for railway alignment. Tiedao Xuebao/Journal of the China Railway Society, 40(3), 6-15.