Article

Research and Implementation of Hybrid Intelligent Wargame Based on Prior Knowledge-DQN Algorithm

Yuxiang Sun [1,*], Bo Yuan [2], Tao Zhang [1], Bojian Tang [1], Wanwen Zheng [1], and Xianzhong Zhou [1,*]

[1] School of Management and Engineering, Nanjing University; sunyuxiangsun@126.com(Y.S.); ztfromchn@163.com(T.Z.);bojiantangnju@163.com(B.T.);18392102175@163.com(W.Z.);zhouxz@nju.edu.cn(X.Z.)

[2] School of Electronics, Computing and Mathematics, University of Derby; b.yuan@derby.ac.uk(B.Y.)

* Correspondence: sunyuxiangsun@126.com; Tel.: +86-1882-705-9351

Abstract: The reinforcement learning problem of complex action control in the Multi-player wargame is a hot research topic in recent years. In this paper,a game system based on turn-based confrontation is designed and implemented with the state-of-the-art deep reinforcement learning models. Specifically, we first design a Q-learning algorithm to achieve intelligent decision-making, which is based the DQN (Deep Q Network) to model the complex game behaviors. Then, a priori-knowledge based algorithm PK-DQN (Prior Knowledge- Deep Q Network) is introduced to improve the DQN algorithm, which accelerates the convergence speed and stability of the algorithm. The experiments demonstrate, the correctness of the PK-DQN algorithm is validated and its performance surpass the conventional DQN algorithm. Furthermore, the PK-DQN algorithm shows effectiveness in defeating the high level of rule-based opponents, which provides promising results for the exploration of the field of smart chess and intelligent game deduction.

Keywords: DQN Algorithm, Policy Modeling, Prior Knowledge, Intelligent Decision

1. Introduction

AlphaGo defeated world go champion Lee in 2016, which caused a new upsurge of AI, especially in the field of game AI. The goal of game AI is not only to let machines play games, but also to simulate the real-world operation through game simulation environment [1]. AI researchers can conduct experiments in games and transfer successful AI capabilities to the real world applications. Although AlphaGo is a milestone of the goal of general artificial intelligence, in general, the problems it represents are still relatively simple compared with the real world [2]. Therefore, in recent years, researchers have focused on the real-time strategy (RTS) games [3][4][5][6], such as Defence of the Ancients (Dota) [7] and StarCraft[8][9], which represent the dynamic game of asymmetric information beyond the Go game.

To master RTS gams, players need to have strong skills in both macro strategic operation and micro execution. In recent studies, micro level implementation has received extensive attention and attempts [10] [11]. So far, OpenAI has made the latest progress in using Dota2 AI (OpenAI Five) developed by reinforcement learning. OpenAI Five is directly trained in micro level action space, using near end strategy optimization algorithm and team reward [12]. OpenAI Five showed more powerful team fighting skills and coordination ability than top professional dota2 team in 2018 international competition [7]. OpenAI approach does not explicitly model macro strategies, but attempts to use micro level game to learn the whole game. However, due to the weakness of AI's macro strategic decision-making ability, OpenAI Five failed to beat the professional teams [13] [14]. Although some achievements have been made in RTS games, there are still many difficulties in both OpenAI 's game AI on DOTA and Deepmind's game AI on StarCraft [15] [16] [17].

On the other hand, a few scholars have studied war chess deduction. Round-robin confrontation is closer to the AI achieved by AlphaGo, and it is easier to solve the problems of micro-strategy modeling and macro-strategy modeling, so it is easier to achieve the AI expected by AlphaGo than RTS games [18] [19] [20] [21].

On this basis, we developed an intelligent game simulation environment, and verified its feasibility through simulated experiments, where a series of basic functional interfaces can be configured to realize the confrontation of different types of operators on different maps. Secondly, this paper uses this platform to verify the feasibility of DQN algorithm in the field of wargames. The DQN algorithm model combined with prior knowledge (PK-DQN) is established, and the DQN algorithm is optimized to realize PK-DQN algorithm. Compared with DQN algorithm, the PK-DQN algorithm has better convergence effect and more stable, and has a higher winning rate in the intelligent war game simulation environment. Finally, via simulations, the PK-DQN algorithm is proved to be able to defeat the high-level rule-based opponents, which provide an important foundation work for futher exploration of deep reinforcement learning in turn-based games.

2. Intelligent decision-making model of tactical chess based on Q-Learning

Through in-depth analysis of the Q-learning method, the intelligent decision-making model of tactical wargame based on Q-Learning is initially implemented to provide basis for the construction of intelligent wargames.

(1) The principles of the Q-learning

Creating a table where you can calculate the expectation of the maximum future reward for each action in each state. Based on this table, we can calculate the best action for each state. Each state (square) allows seven possible operations: East, West, Northeast, Northwest, Southeast, Southwest, and Still; 0 represents an impossible movement.

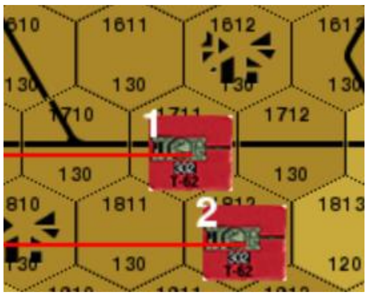
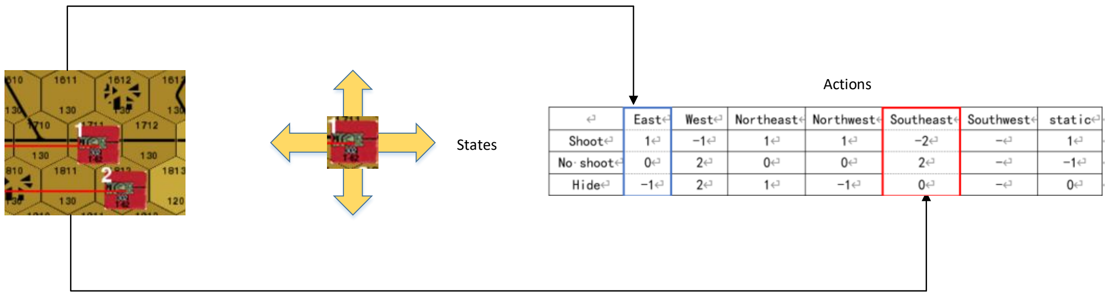


Figure 1. Schematic diagram of operator movement

During the calculation, we can convert the grid into a table.



	East	West	Northeast	Northwest	Southeast	Southwest	static
Shoot	1	−1	1	1	−2	−	1
No shoot	0	2	0	0	2	−	−1
Hide	−1	2	1	−1	0	−	0

Figure 2. Transformation of q-table

This form is called q-table (Q stands for the "mass" of the action). Each column will represent seven operations (East, West, Northeast, Northwest, Southeast, Southwest, and Stationary), with rows representing states. The value in each cell represents the maximum future reward expectation for a given state and a corresponding action. The score of each q-table will represent the maximum expectation of future reward obtained by taking corresponding actions under the given optimal strategy. Each value in the q-table is obtained using the Q-learning algorithm.

The action value function (or Q function) has two inputs: state and action. It will return for future reward expectations for performing the action in that state.

$$Q_{\pi}(s,a)=E_{\pi}[r_t+1+\gamma r_t+2+\gamma 2r_t+3+\ldots|A_t=a,S_t=s]$$

The general idea of intelligent decision-making modeling of tactical chess based on Q-Learning is described above, but there are still significant problems in using the algorithm of Q-learning to build intelligent chess. Although Q-learning method can provide basic ideas for the construction of intelligent wargame, the traditional table form stores state and corresponding Q-value to find the best executive action. But in the case of too complex state, the computer memory is limited, it is difficult to achieve all storage, and it is also time-consuming work to search all tables. Therefore, this paper further considers whether the improved DQN algorithm can be used to realize the behavior decision-making modeling of intelligent wargame.

(2) DQN based behavior decision-making model of intelligent wargames

In this paper, we adopted the Q-learning to find big problems in the process by using the DQN algorithm to realize the behavior decision-making modeling for intelligent wargames. After analyzing DQN algorithm and building the intelligent wargame model of DQN algorithm, we successfully implemented the intelligent wargame based on the DQN algorithm.

DQN is an intelligent algorithm which combines deep learning and reinforcement learning. It mainly uses deep learning to take the state and action as the input value of neural network. After the analysis of neural network, it gets the Q-value, so it is unnecessary to record the Q-value in the table, and directly use a neural network to generate the Q-value. Then, using the correct Q value provided by reinforcement learning and the Q-value to calculate the loss function, so as to improve the network weight of deep learning models. According to the output action value of the neural networks and the principle of Q-learning, the method directly selects the action with the maximum value as the next action to be done. It integrates the neural network and Q-learning method, namely the DQN (deep-q network) algorithm.

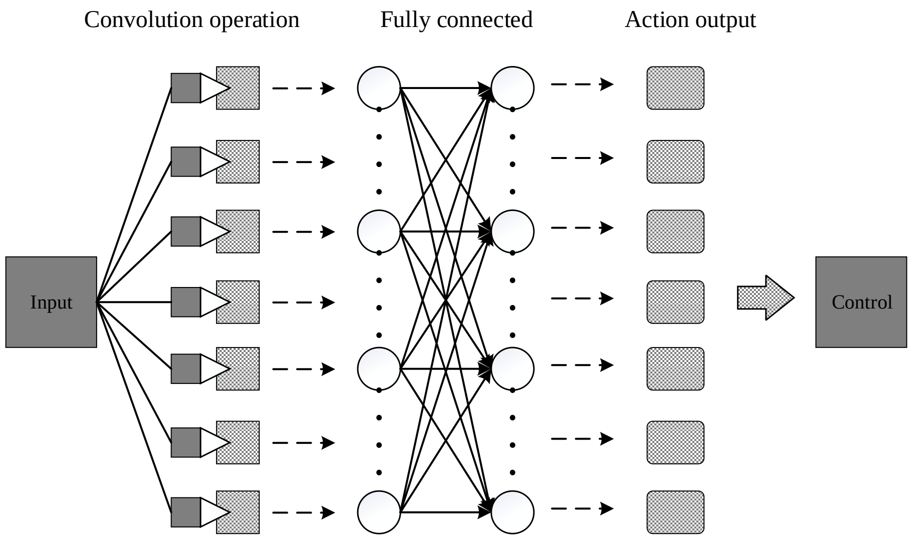


Figure 3. Algorithm framework of deep Q neural network

The algorithm is as follows:

Algorithm 1: Deep Q-learning with experience in reply.

```
Initialize reply memory D to capacity N
Initialize action-value function Q with random weights $\theta$
Initialize target action-value function Q with weights $\theta^{-} = \theta$
For episode = 1, M do
    Initialize sequence $s_1 = \{x_1\}$ and preprocessed sequence $\phi_1 = \phi(s_1)$
    For $t = 1$, T do
        With probability $\varepsilon$ select a random action $a_t$
        Otherwise select $a_t = argmax_a Q(\phi(s_t), a; \theta)$
        Execute action $a_t$ in emulator and observe reward $r_t$ and image $x_{t+1}$
        Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$
        Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in D
        Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from D
        Set $y_j = \begin{cases} r_j, \text{if episode terminates at step } j+1 \\ r_j + \gamma max_{a'} \widehat{Q}(\phi_{j+1}, a'; \theta^{-}), \text{otherwise} \end{cases}$
        Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$ with respect to the network parameter $\theta$
        Every c steps reset $\widehat{Q} = Q$
    End For
End For
```

Figure 4. DQN Algorithm

In this paper, we utilize the DQN algorithm to implement the operator's action selection, which is to set an RL_Brain to select actions, RL_Brain is a brain that deduces relative to the entire Smart Warfare game. It is set up as a two-layer network, with ten neurons in one layer and the ability to output actions. The action selection of the red side obtains the XY coordinates of the rank of the red side, which is then passed into the obversation as an observation input array. RL_Brain trained neural network selects the maximum Q-value action, decides the output state action space value, calls the corresponding movement function or shooting function after determining the value, and then determines the next action.

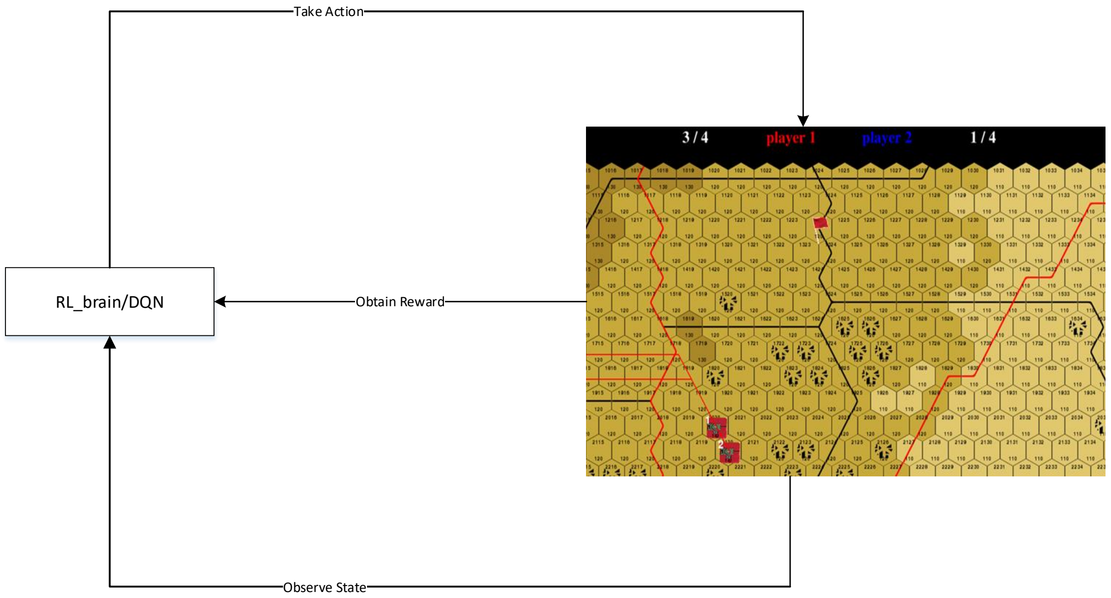


Figure 5. Action selection of intelligent wargame

3. Construction of Intelligent Chess Model Based on PK-DQN Algorithm

How to effectively make, evaluate and choose the action plan is a challenging problem that restricts the intelligent decision system to enter the practical application. To solve this problem by traditional methods, it is necessary to model the environment and rules reasonably. However, in many aspects of modeling, human subjective factors are inevitably introduced, which greatly reduces the accuracy and rationality of the final decision-making scheme. Therefore, we break through the traditional method of action plan reasoning based on model knowledge, and explore the technical solution combining the experience judgment and intuitive reasoning that fit the decision-making thinking of commanders.

As one of the methods to solve sequential decision making, reinforcement learning (RL) has been predominantly combined with deep learning in the field of artificial intelligence in recent years, which has achieved remarkable results and become a representative machine learning method to break through cognitive intelligence. This mechanisms and methods of reinforcement learning can be used as the key technology of simulation, deduction and evaluation of the scheme, because it accords with the decision-making thinking mode of commanders for complex combat problems.

This paper introduces DQN algorithm in reinforcement learning for intelligent Red and Blue Army in military chess deduction. Red AI is built with DQN algorithm, Blue AI is built with rules. Red operator based on DQN algorithm is trained with rules-based blue algorithm through red-blue game, which provides more realistic simulation scenarios and solves battle tactics. The practical problem of low AI level in military chess deduction assists command and decision makers in accurately understanding, rational thinking and quick decision-making of battlefield situation. Based on the analysis of state space and action space of chess deduction, the feasibility study is carried out by using DQN algorithm of reinforcement learning. The state space of chess deduction can be defined as position coordinates X and Y, and the real-time state (maneuver, concealment, design) of operators can form the state space of chess deduction. The action space of chess deduction can also be defined as specific and limited operation. Therefore, this section first describes the design of smart chess system, then presents the expert knowledge model in the field of dispatching chess, and builds the action strategy model based on DQN algorithm, and finally establishes the process model of integrating prior knowledge with DQN algorithm i.e., the PK-DQN algorithm.

4.1 Design of intelligent wargame system

The red square operator of the whole intelligent wargame system inputs at a certain time (maneuver, concealment, design) and XY coordinate, while the terrain reads the excel table within the engagement range, and the terrain hexagon lattice mainly includes high-level and label. The whole deduction environment is based on this system setting.

The whole game environment is mainly composed of four modules, including learning module, simulation environment module, distribution module and memory module. The relationships among the modules are shown in the following Fig.6:

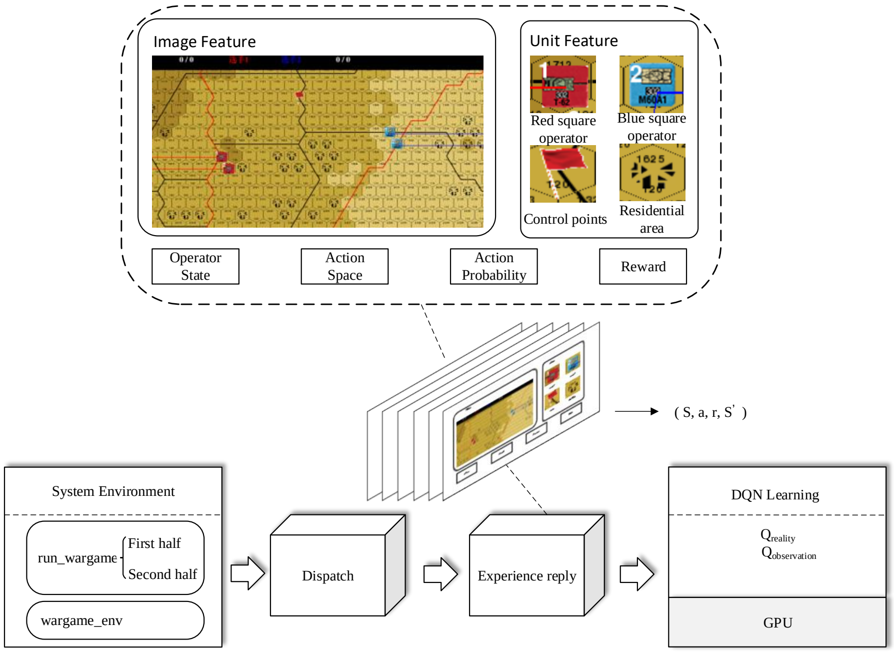


Figure 6. AI Server implements how the AI model interacts with the environment. The Dispatch Module is a station for sample collection, compression and transmission. The Memory Pool is the data storage module, which provides training instances for the RL Learner. Note that these modules are decoupled and can be flexibly configured, so that our researchers can focus on the algorithm design

and the logic of the environment. Such a system design is also applicable to other multi-agent competitive problems. The details of these modules are provided as follows:

Learning module: Learning module is the core part of intelligent algorithm. By calling status, action, return value and next step status in batch size, neural network parameters are constantly updated. In order to reduce the cost function value efficiently, the learning module and memory module use a memory structure together. The trained model synchronizes to the deduction environment module in the form of point-to-point quickly, and then realizes the selection of the model of the action.

Deduction environment module: The deduction environment module consists of the interaction between the deduction environment and AI model, as well as the basic functions of defining the whole deduction environment. The functions comprise judging the winning conditions, checking the number of rounds, reading in the deduction scenario, etc. Save the basic parameters related to the deduction, including both sides score, both sides kill number, both sides survival number and both sides win number. The whole environment generates a confrontation environment of red side and blue side, and sets a clear number of rounds, which enables playing a game within the number of rounds.

Allocation module: The allocation module is responsible for collecting the sample data obtained from each step from the deduction environment module, including the current status, return value, action and next step status, and passing the collected data into the memory module in the form of array.

Memory module: Memory module is a part of memory space, which is used to set the size of memory space, and then transfer the allocated array data into and store them in turn. When the storage space is larger than the memory space, the data before is removed, and the data of batch size are also extracted to the learning module to update the strategic network, so as to reduce the loss function.

4.2 Prior knowledge of intelligent wargame

In this paper, it is found in the training process of intelligent wargame that only relying on DQN algorithm for training; there is a large randomness for the cold start of the start, and the overall convergence speed is slow. Therefore, this paper considers introducing the concept of prior knowledge. By adding prior knowledge to the process of action selection, it can prevent the algorithm from falling into local optimum, reduce invalid exploration and improve the learning efficiency of the algorithm. Priori knowledge considered in this paper is divided into two parts. The first part is the basic function realization of intelligent wargame, including the basic definition of wargame action. Another part is the prior knowledge based on the experience of domain experts, which can accelerate the convergence of the training process.

4.2.1 Prior knowledge of basic functions of wargame

The design of intelligent wargame must be based on the corresponding basic functions. Through invoking of basic functions, the algorithm of intelligent wargame can be realized, and finally the establishment of intelligent engine can be realized. Its main functions are as follows:

(1) Move function

Initialize the starting position, assign the value in the scenario, calculate the X, Y coordinates of each operator, obtain the coordinates of the surrounding hexagonal lattice, and then select a coordinate to assign the value in the acquired hexagonal lattice coordinates, and then move the coordinates, including the East, West, Northeast, Northwest, Southeast, Southwest six directions.

(2) Integral function of shooting reward

Shoot the enemy operator, obtain the coordinates of the enemy operator, and then judge whether the enemy operator exists after shooting. If it exists and the coordinates correspond to the coordinates of the enemy operator, the corresponding reward score will be obtained, otherwise, no score will be obtained.

(3) Shooting function

Obtain the coordinate position of the operator, judge whether the enemy operator can be observed by calling the visual function, if the observed distance can be shot, set the strike effect according to the enemy and distance.

(4) Get adjacent coordinate function

Input the X and Y coordinate of the operator to represent the coordinates of the hexagonal lattice, and output the list to represent the coordinates of the surrounding hexagonal lattice in the form of a list.

(5) Query the distance between two hexagonal lattices

Input X0, Y0, X1, Y1 as the coordinates of int, indicating the coordinates of the hexagon lattice at the beginning and the hexagon lattice at the end, and output as the distance between the two hexagon lattice.

(6) Get operator state information function

The current coordinates of the operator and the maneuvering state of the turn are obtained by the function.

(7) Check whether the operator can observe the opposite operator

Input the state information of the opposite operator, observe that the output of the opposite operator is true, but not false.

The rule of the whole intelligent wargame is that red and blue fight each other, and the operators of both sides can move, hide, direct aim and indirect aim. In this way, mobile means to input X and Y coordinates, which represent the coordinates of adjacent hexagons, output effect, and move the operators. ‘Hide’ means to ensure that the operators enter the concealed state, which is not conducive to being attacked. The direct aim is to input enemy operators, output phase Should shoot effect, shoot enemy operator; input X, Y represents target hexagonal grid coordinate, output effect, indirect aim target hexagonal grid.

4.2.2 Prior knowledge of domain experts

This paper designs the prior knowledge of the experts in the field of intelligent wargame, which mainly includes the moving strategy and shooting strategy of the intelligent wargame operators. At the same time, this paper also constructs the sub strategy of the professional players in the process of the competition based on the data of the wargame competition, forms a static comprehensive potential energy table, and realizes the prior knowledge of the experts in the field through the full simulation, combining the prior knowledge of the experts in the field with DQN calculation Method to accelerate the convergence of loss function. The designed operator action strategy based on prior knowledge of domain experts is shown in Fig. 7:

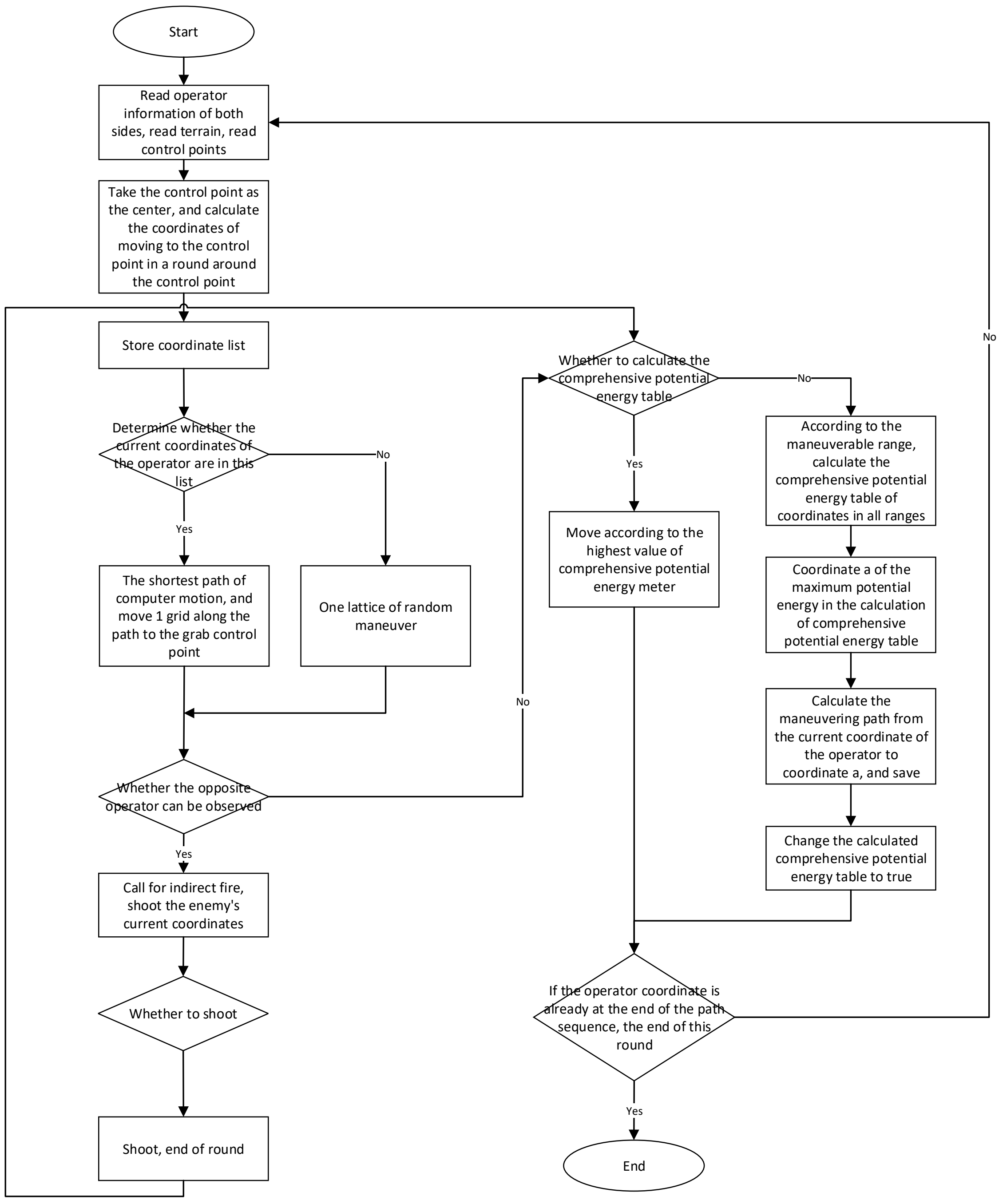

Start
Read operator information of both sides, read terrain, read control points
Take the control point as the center, and calculate the coordinates of moving to the control point in a round around the control point
Store coordinate list
Determine whether the current coordinates of the operator are in this list
No
Yes
The shortest path of computer motion, and move 1 grid along the path to the grab control point
One lattice of random maneuver
Whether the opposite operator can be observed
Yes
Call for indirect fire, shoot the enemy's current coordinates
Whether to shoot
Shoot, end of round
No
Whether to calculate the comprehensive potential energy table
No
Yes
Move according to the highest value of comprehensive potential energy meter
According to the maneuverable range, calculate the comprehensive potential energy table of coordinates in all ranges
Coordinate a of the maximum potential energy in the calculation of comprehensive potential energy table
Calculate the maneuvering path from the current coordinate of the operator to coordinate a, and save
Change the calculated comprehensive potential energy table to true
If the operator coordinate is already at the end of the path sequence, the end of this round
No
Yes
End

Figure 7. Flowchart of Domain Expert Knowledge

First, read the location, terrain and contention point information of both sides. Then, taking the point of contention as the center, the coordinates around the point that can maneuver to the control point in a round are calculated, and the coordinate list is stored. Determine whether the current coordinates of the operator are in this list. If so, the shortest path of the computer movement, the soldier moves 1 grid along the path to the point of contention. Judge whether the opponent's algorithm can be observed, and if so, prepare for long-range shooting. Then judge whether it meets the shooting conditions. If so, it will strike the enemy's current coordinates. If not, judge whether the comprehensive potential energy table has been calculated. If not, according to the maneuverable range and the comprehensive potential energy of the coordinates within the moving range of the computer, move to the maximum potential energy coordinate. If the potential energy is in the same direction, the killing potential energy is compared, and then the maneuver path from the current coordinate to the coordinate a is generated, which is called repeatedly. Finally, the turn ends when it reaches the end of the path sequence and moves to the next coordinate of the path sequence.

4.3 Action strategy based on DQN algorithm

(1) Policy space description

In order to realize the intelligent wargame based on DQN algorithm, the description of operator's state space and action space should be made clear. The state space mainly includes the X and y position coordinates of the deduction scenario, and the number of values is 30 * 30. The action parameters of the operator mainly include maneuver, shooting and concealment, as shown in Table 1.

Table 1. State space description of tactical command decision

State space description of tactical command decision		
State parameters	**Position coordinate X**	**Position coordinate y**
Dimension	1	1
Number of values	30	30

Table 2. Description of tactical action space

Tactical action space description		
Action parameters	**Move (Six corners)**	**Shooting**
Dimension	1	1
Number of values	7	3

Among them, the maneuverable directions of the operator are 7 states of East, West, northeast, northwest, Southeast, southwest and static, which are defined as 0 ~ 6 respectively, t, as shown in Table 2. The firing states of the operator in one cell are shooting, not shooting and concealing.

(2) Reward function design

The reward function is mainly divided into distance reward and shooting reward. Distance reward is used to obtain R1 reward values for the winning control point. The European distance between the route selection operator and the winning control point is set as the greater the European distance from the winning control point, the lower the reward value. If the winning control point is obtained, the return value is set as a larger value. Finally, the observation values x, y coordinates, actions and return values of each step are stored in the transition for later recall study.

$$\text{reward} = \text{reward} - \text{distance}^2$$

If the shooting reward is to shoot and hit the enemy operator, R1 rewards will be given, otherwise no reward value will be given.

(3) Path planning

According to path planning set by reward, the shortest distance from the control point will be automatically found. In the early stage of training, the path selection will appear in the repeated movement of a point, but in the late stage of training, we will see the red square operator and gradually find the closest distance to the control point. As shown in the figure below, the red line behind the red square operator is the trajectory of the operator's movement. The operator has learned that the best way to reach a hold point.

(4) Shooting

In this paper, the reward value is to define the effect of shooting. Due to the large shooting space, the training convergence will be too slow. Therefore, for the shooting part, the shooting rules are integrated to speed up the convergence of the training. The shooting rules mainly include two ideas: whether it can be intervisibility, and if it can be intervisibility, direct shooting. Another idea is to judge the distance between all operators of both sides and calculate the nearest distance between them. According to the actual combat operation of professional players, when the distance between operators is 12, the shooting effect is the best. Therefore, 12 is taken as the mark point. If the distance is greater than 12, the operator moves towards the winning control point continuously. If the distance between operators of both sides is less than or equal to 12, and an enemy operator remains still. The state can be used for shooting, otherwise the operator moves to the nearest distance of the control point and fire.

4.4 DQN algorithm model combined with prior knowledge

According to the introduction of prior knowledge, this paper defines the prior knowledge as follows:

(1) Let the state sequence of DQN algorithm be S, and its formula be:

$$S=\left[s_1,s_2,s_3,...,s_i\right],i=1,2,3...$$

(2) Set the action sequence as A, and its formula is:

$$A=\left[a_1,a_2,a_3,...,a_i\right],i=1,2,3...$$

(3) Let the characteristic state sequence of prior knowledge strategy be P, and its formula be:

$$P=\left[p_1,p_2,p_3,...,p_i\right],i=1,2,3...$$

When the characteristic state p_i appears, the optimal action selection is a^*, then the state is set as the characteristic state p^*. The mapping relation $p^*\to a^*$ between the characteristic state and the optimal action is established. The rules formed by this mapping relationship are called prior knowledge. Among them, the state sequence S is the input of the algorithm, the characteristic state P may be the input of the algorithm or other signals, and the characteristic state and the state sequence satisfy $P\cap S=\varnothing$.

(4) According to the concept of Q value in DQN algorithm, the prior Q value vector is defined as Q_p, and the Q_p formula is

$$Q_p = \left(Q_{pa_1}, Q_{pa_1}, Q_{pa_1}, \ldots, Q_{pa_i}\right)^T$$

Where Q_{pa_i} is the prior Q value of action a_i, $0 < Q_{pa_i} < 1$, and vector Q_p is the set of prior Q values of all actions when the prior knowledge is known.

(5) Because there are many factors that affect the result of decision-making, and the prior knowledge has only partial influence on the decision-making of action selection, therefore, the influence of prior knowledge on action selection is not completely determined. According to the influence degree of prior knowledge on action selection, this paper sets μ as the influence degree of prior knowledge on action selection. When the feature state p^* appears and the prior knowledge is completely related to the selection action a^*, $\mu = 1$. $0 < \mu \leq 1$, there is no prior knowledge that has no effect on action selection.

(6) According to the above definition, after the prior knowledge is added, the final action selection Q value is Q_F. first normalize Q, then calculate Q_F, and the Q_F calculation formula is:

$$Q_F = \mu Q_p + (1 - \mu) Q \quad (14)$$

Where Q is the Q value vector selected by DQN algorithm according to the action estimated by the current state. When the characteristic state p^* appears, the Q_F value of each action is calculated according to formula 14, and the action is directly selected according to the maximum Q_F value; when the characteristic state p_i does not appear, the Q value of the action is estimated according to DQN algorithm, and then the action is selected according to $\varepsilon - greedy$ rule. There are three main advantages: firstly, by introducing prior knowledge, when the feature state p^* is detected, it can affect the action selection process according to the prior knowledge $p^* \rightarrow a^*$, reducing unnecessary exploration; second, the introduction of prior knowledge does not affect the training process of the algorithm; finally, when there is no feature state, it can judge the action selection through the algorithm to fully explore the state space. The introduction process of prior knowledge is shown in the Fig.8.

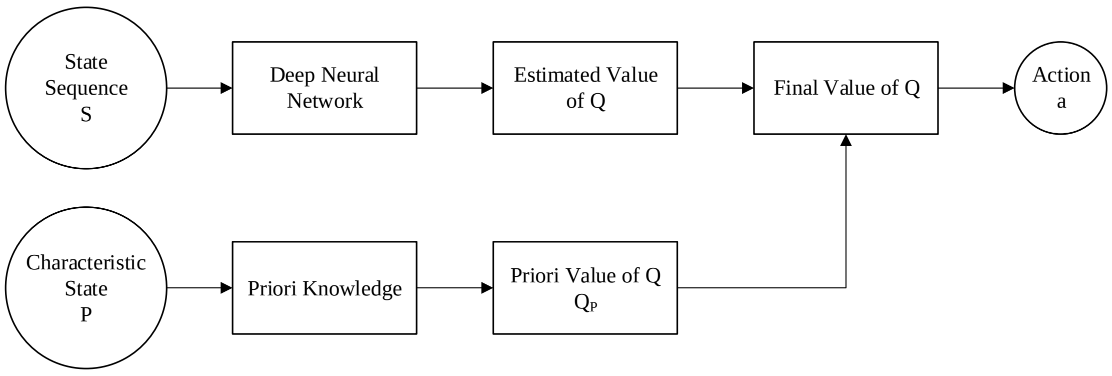


Figure 8. Fusion process of prior knowledge and DQN algorithm

By introducing prior knowledge and intervening in the action selection process of DQN algorithm, valuable information can be fully used during algorithm training. Prior knowledge such as domain expert knowledge, feature signal and classification feature can be added to algorithm training, so that the improved model can effectively avoid invalid exploration. The improved algorithm improves the convergence speed and accuracy of the algorithm, which can effectively improve the training efficiency and save the training cost. During the training process, PK-DQN algorithm selects actions and controls the action selection process according to whether there is characteristic state. According to the definition of priori knowledge, priori Q value and μ in the above formula, the priori knowledge is introduced into DQN algorithm to build PK-DQN algorithm, and PK-DQN algorithm is shown in the Fig.9.

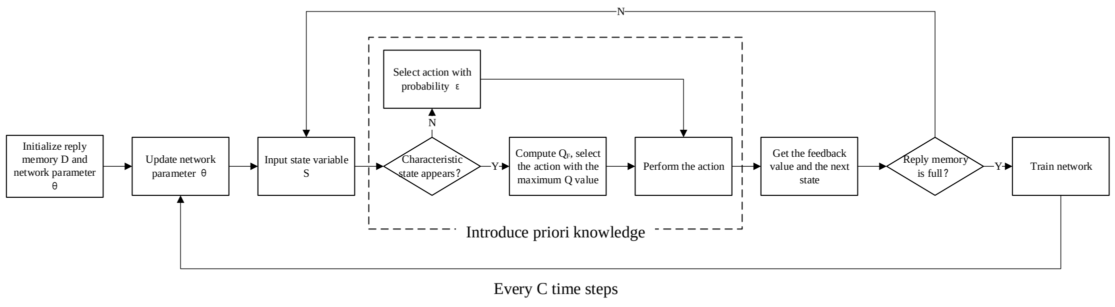

Initialize reply memory D and network parameter θ
Update network parameter θ
Input state variable S
Select action with probability ε
N
Characteristic state appears?
Y
Compute Q_F, select the action with the maximum Q value
Perform the action
Get the feedback value and the next state
Reply memory is full?
Y
N
Train network
Introduce priori knowledge
Every C time steps

Figure 9. DQN algorithm model combined with prior knowledge

Algorithm: Prior Knowledge-Deep Q Network (PK-DQN) is combined with prior knowledge. Input: state s, output: Q neural network weight.

Step 1: Initialize experience value d with capacity n

Step 2: Initialize Q neural network and set the random weight as θ;

Step 3: At the beginning of iteration t = 1, input the initial state s_t;

Step 4: If the characteristic state p^* appears, calculate the Q_F value from the formula, and select the action a^* with the largest Q_F value to output; otherwise, output the action a_t according to the $\varepsilon - greedy$ rule;

Step 5: Executes action a^* or a_t;

Step 6: Obtains the feedback r_t value and s_{t+1};

Step 7: Store (s_t, a_t, r_t, s_{t+1}) in the experience pool;

Step 8: Judge whether the data amount in the experience pool reaches n. if it reaches n, train Q neural network, and update the weight θ of target Q neural network after constant C time steps; if it does not reach n, repeat step 3-step 8. When the value θ is less than the threshold value, the algorithm is considered to reach the convergence state.

The above figure is the detailed flow chart of the algorithm, in which the dotted line part indicates the introduction of prior knowledge into DQN algorithm. Next, through specific experiments to verify the improvement effect of PK-DQN algorithm.

5. Experimental verification

In this paper, the training environment is based on the intelligent game simulation environment independently designed and developed by our team. PK-DQN algorithm and DQN algorithm control operators are respectively used for confrontation, including independent flag grabbing and mutual shooting. The stability and convergence rate of the two algorithms are compared to verify the influence of the introduction of prior knowledge on the training effect of the algorithm.

5.1 Experimental platform

In order to verify the effectiveness of PK-DQN algorithm, this paper uses the self-developed simulation environment as a test example, and the main configuration of the experiment is listed as follows:

(1) Win 10: operating system, various task processing.

(2) CUDA v8.0: GPU C language library. Calculate the same device architecture

(3) cuDNN (v6.0.21): deep learning primitive Library Based on CUDA

(4) Tensorflow (v1.0): a deep learning framework developed by Google

The whole environment includes DL server and AI server. AI server is configured with DQN algorithm and PK-DQN algorithm for data training, which is transmitted to DL server after training to form deep neural network model. The cloud server transmits the original confrontation data and action sequence to the AI server, and receives the action sequence and cognitive model processed by the AI server. The cognitive model and action sequence formed are further transmitted to the intelligent decision-making system, and then to the experimental environment for simulation and deduction through decoupling. The experimental environment obtains new data through deduction, which is aggregated and then transmitted to the intelligent decision-making system. The intelligent decision-making system retransmits the new data, AI configuration and action sequence to the cloud server for a new round of training.

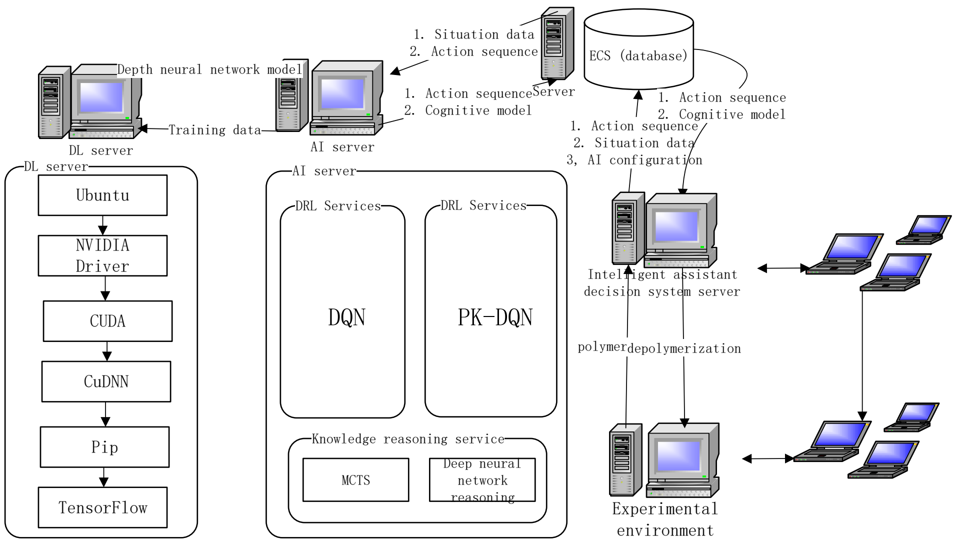


Figure 10. Based on DL server, AI server, simulation environment, intelligent assistant decision system and database environment, the architecture of intelligent wargame system is constructed

Scenario Description:

In this experiment, two operators, red and blue, are supposed to fight in the environment. The winning condition is to seize the red flag to seize the control point or to destroy all the operators of the enemy. At present, the urban residential area map is used, and the terrain of the main battlefield area is shown in the figure below.

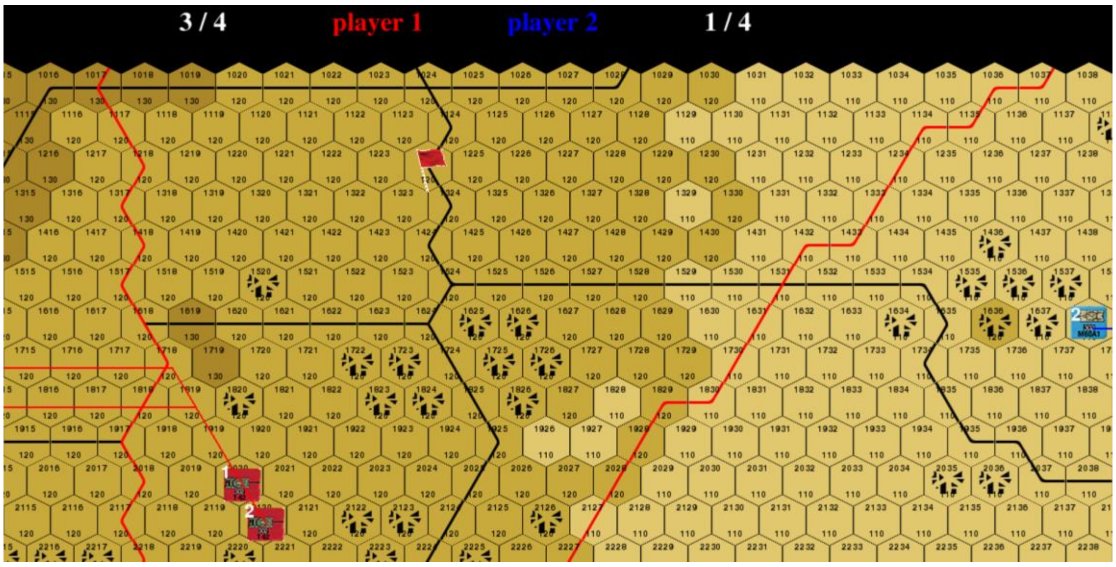


Figure 11. Scenario Description: red and blue have two operators (tanks) to fight each other. The red flag in the middle is the key point. The deeper the color is, the higher the elevation is. According to the grid of urban residential areas, it is indicated in black, which is conducive to concealment. The black line represents the first class road and the red line indicates the second class road. The operator (tank) has different moving speed on the first road and the second road respectively.

The details are as follows:

(1) Elevation: elevation information is represented by color depth. There are five different colors to represent five different elevations. Elevation gradually increases with color from light to depth, and the height difference between adjacent changing colors is 20 meters.

(2) Operators: two for red and two for blue, each of which represents platoon, the minimum resolution unit of the Army Armored synthetic battalion. The attributes describing the operators mainly include two categories: tactical command decision state space and tactical action space.

Consumption: since the movement of the operator will consume oil, the initial oil quantity is set in the environment. The oil quantity will be consumed every time the operator moves a grid; the oil consumption varies with different elevations, and the higher the elevation, the greater the oil consumption.

(3) The map size is 66x51 and the map number is 83.

(4) There is a total of 1 control point, which are: control point, coordinate 80048, score 80;

(5) There are two pieces in the red and blue sides respectively. The initial position and terrain of the two tanks are described as follows:

Table 3. Initial situation of opposing parties

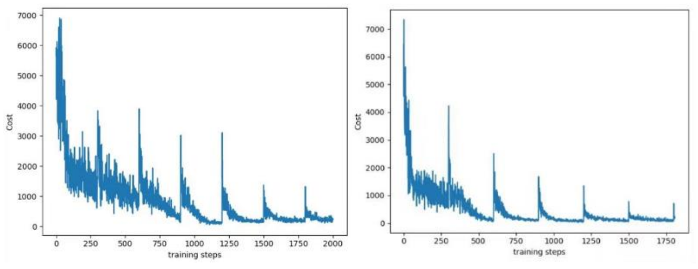
Initial position	Red	Blue	Terrain
Tank1	80015	90015	Plain
Tank2	70081	80081	Plain

5.2 Experimental results and analysis

In this experiment, PK-DQN algorithm is applied to the self-developed confrontation environment, and the performance of the algorithm is tested and compared with DQN algorithm.

Through the comparative analysis of the experimental results, the experimental results are shown in Figure 12. The horizontal axis in the figure represents every step of training. The vertical axis loss represents the size of the cost function $L_i(\theta_i)$.

(1) In terms of convergence speed, DQN algorithm needs about 1500 steps to reach the convergence threshold of the algorithm, which is still not less than 1000, while PK-DQN algorithm can reach the convergence threshold of the algorithm in 1500 steps. Compared with DQN algorithm, PK-DQN algorithm is faster and more stable. See Figure 17 for details. The reason is that by introducing prior knowledge, PK-DQN algorithm reduces the number of times of exploration in the training process, speeds up the optimization speed of Q neural network, and the algorithm converges rapidly with little fluctuation, so it can reduce the intervention of prior knowledge in the action selection process after the algorithm converges, and at the same time ensure the training efficiency and stability of the algorithm.



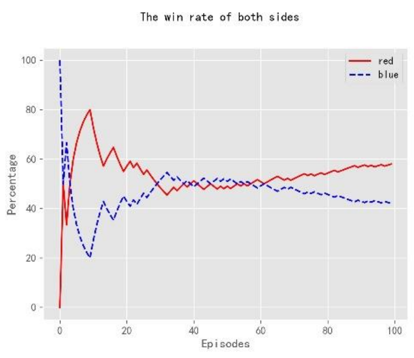
(a) **(b)**

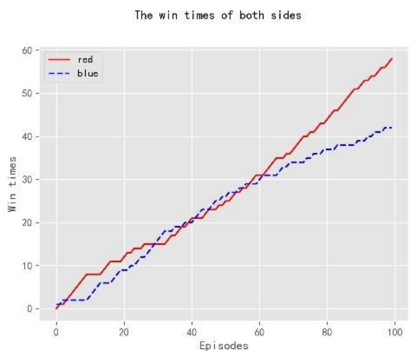
Figure 12. **(a)** Performance of DQN learning curve in simulation environment; **(b)** Performance of PK-DQN learning curve in simulation environment. The X coordinate represents the number of training steps, that is, the total steps of confrontation training; the Y coordinate represents the size of the cost curve value $L_i(\theta_i)$.

(2) In the aspect of algorithm winning rate, To verify the faster convergence of PK-DQN, we guarantee that PK-DQN and DQN will be trained together for 24 hours at the same time, then PK-DQN and DQN will be used as red AI and rule-based Blue AI to compete against each other, and then observe the winning rate of the two algorithms compared with rule-based Blue AI. The winning rate of DQN algorithm and PK-DQN algorithm is 57% and 67% respectively. It shows that the introduction of prior knowledge can obviously improve the intelligence of operators. The specific winning games are shown in the table below.

Table 4. Comparison of the number of winning matches between red and blue teams after swapping positions

Algorithm	**Victory number**	**Rounds**
DQN	57	100
Rule	43	100



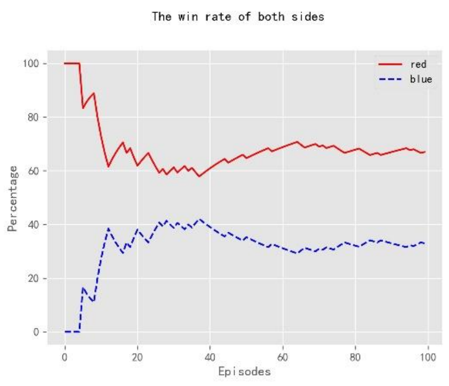


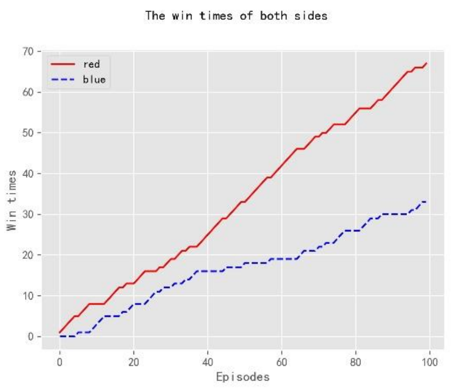
(a) **(b)**

Figure 13. (a) Win rate: The red side is the AI of DQN intelligent algorithm and the blue side is rule-based AI; **(b)** Win times: The red side is the AI of DQN intelligent algorithm and the blue side is rule-based AI; The winning rate and the number of wins for the red and blue sides. The first round wins so one side starts from 1 and the other from 0.

Table 5. Comparison of winning matches between red and blue

Algorithm	**Victory number**	**Rounds**
PK-DQN	67	100
Rule	33	100





(a) **(b)**

Figure 14. (a) Win rate: The red side is the AI of PK-DQN intelligent algorithm and the blue side is rule-based AI; **(b)** Win times: The red side is the AI of PK-DQN intelligent algorithm and the blue side is rule-based AI; The winning rate and the number of wins for the red and blue sides. The first round wins so one side starts from 1 and the other from 0.

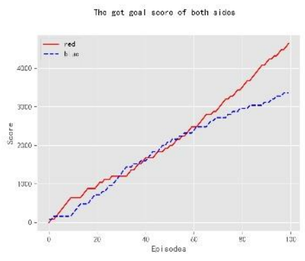
(3) In terms of algorithm efficiency. To verify the efficiency of training, we use the 80% winning rate as the verification standard to see how long it takes for PK-DQN and DQN to reach 80% winning rate. Both algorithms are against the rule-based blue AI. It takes 23 hours and 17 minutes for PK-DQN to be stable to win rule-based AI in the confrontation, and 32 hours and 39 minutes for DQN algorithm alone. It further shows that the introduction of prior knowledge can significantly accelerate

the convergence speed of the algorithm. The experimental results are shown in Table 6, where T is the total training time and N is the number of operators (tanks) in the antagonism.

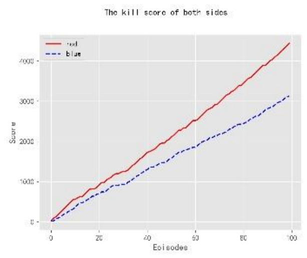
Table 6. Stable winning training time comparison

Algorithm	**T（Total Time）**	**N（Number）**
DQN	32h39min	4
PK-DQN	23h17min	4

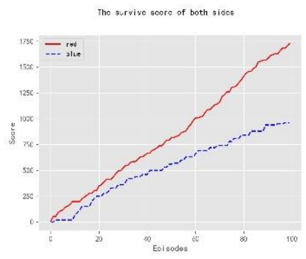
The experimental results show that the PK-DQN model can reduce the number of times to explore during training, and improve the problem that the DQN algorithm takes too long to train. It shows that the introduction of prior knowledge improves the performance of DQN algorithm, and has a certain theoretical significance for improving the efficiency of the algorithm.



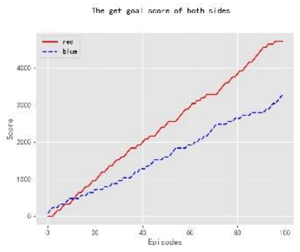
(a)



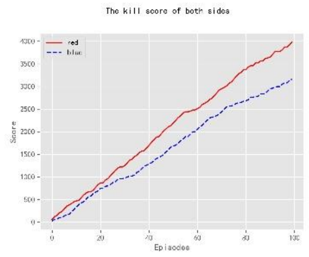
(b)



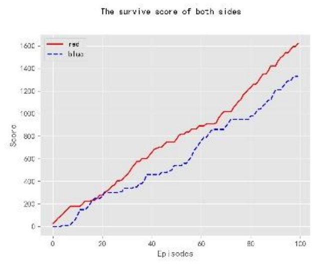
(c)



(d)



(e)



(f)

Figure 15. (a) The get goal score of both sides(Red: DQN); **(b)** The kill score of both sides(Red: DQN); **(c)** The survive score of both sides(Red: DQN); **(d)** The get goal score of both sides(Red: PK-DQN) **(e)** The kill score of both sides(Red: PK-DQN) **(f)** The survive score of both sides(Red: PK-DQN). The x-axis is the training episodes, and the y-axis is the score. Red and blue represent two teams in the confrontation environment.

6. Conclusion

The main innovative work of this paper is to independently develop a smart chess deduction environment, which has been verified by experiments and configured with a series of basic functional interfaces to achieve the confrontation of different types of operators on different maps. It can provide validation and effect analysis for subsequent large-scale and different kinds of war-chess combat deduction experiments. At the same time, this paper uses this platform to verify that DQN algorithm can be applied in the field of Chess of War, to achieve the deduction of Smart Chess, and to explicitly verify that DQN algorithm can defeat high-level rule-based opponents, which provides the first step of work verification for the exploration of the field of Smart Chess and the deduction of Smart Game.

At the same time, it is found that DQN training is difficult to converge for a long time during the training process, so a priori knowledge is introduced to improve the DQN algorithm, and a PK-DQN algorithm is proposed. The experimental results show that the PK-DQN model can reduce the number of times to explore during training, and improve the problem that the DQN algorithm takes too long to train. It shows that the introduction of prior knowledge improves the performance of DQN algorithm, and has a certain theoretical significance for improving the efficiency of the algorithm. Subsequently, a series of work will be done on this platform, including the experimental validation of DDQN algorithm, A3C, PPO algorithm and a series of improved algorithms in the deductive antagonism of Smart Chess, as well as the study of collaboration between multi-agents.

7. Patents

The experimental platform used in this paper has declared the copyright of computer software. (grant number 2020R11S0628066).

Author Contributions: Project administration, B.T.; Supervision, X.Z. and B.Y.; Writing—original draft, Y.S.; Writing—review & editing, W.Z and T.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported in part by the Innovation and Creativity Research Program for Doctoral Students of Nanjing University (grant number CXCY19-19).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. De Loura, Mark A., ed. Game programming gems 2. Cengage learning, 2001.
2. Silver, D.; Huang, A.; Maddison, C. J.; Guez, A.; Sifre, L.; Van Den Driessche, G.; Schrittwieser, J.; Antonoglou, I.; Panneershelvam, V.; Lanctot, M.; et al. 2016. Mastering the game of go with deep neural networks and tree search. nature 529(7587):484–489.
3. Nascimento Silva, V., and Chaimowicz, L. 2015. On the development of intelligent agents for moba games. In Computer Games and Digital Entertainment (SBGames), 2015 14th Brazilian Symposium on, 142–151. IEEE.
4. Synnaeve, G., and Bessiere, P. 2011. A bayesian model for rts units control applied to starcraft. In Computational Intelligence and Games (CIG), 2011 IEEE Conference on, 190–196. IEEE.
5. Tian,Y.;Gong,Q.;Shang,W.;Wu,Y.;and Zitnick, C. L. 2017. Elf: An extensive, lightweight and flexible research platform for real-time strategy games. In Advances in Neural Information Processing Systems, 2656– 2666.
6. Wender,S.,andWatson,I. 2012. Applying reinforcement learning to small scale combat in the real-time strategy game starcraft: Broodwar. In Computational Intelligence and Games (CIG), 2012 IEEE Conference on, 402–408. IEEE.
7. OpenAI. 2018a. Openai blog: Dota 2. https://blog.openai.com/dota-2/ (17 Apr 2018).
8. Tian,Y.;Gong,Q.;Shang,W.;Wu,Y.;and Zitnick, C. L. 2017. Elf: An extensive, lightweight and flexible research platform for real-time strategy games. In Advances in Neural Information
9. Vinyals, O.; Ewalds, T.; Bartunov, S.; Georgiev,P.;Vezhnevets,A.S.;Yeo,M.;Makhzani,A.;Küttler, H.; Agapiou, J.; Schrittwieser, J.; et al. 2017. Starcraft ii: a new challenge for reinforcement learning. arXiv preprint arXiv:1708.04782.
10. Synnaeve, G., and Bessiere, P. 2011. A bayesian model for rts units control applied to starcraft. In Computational Intelligence and Games (CIG), 2011 IEEE Conference on, 190–196. IEEE.
11. Wender,S.,andWatson,I. 2012. Applying reinforcement learning to small scale combat in the real-time strategy game starcraft: Broodwar. In Computational Intelligence and Games (CIG), 2012 IEEE Conference on, 402–408. IEEE.
12. Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; and Klimov, O. 2017. Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347.

13. Vincent, J. 2018. Humans grab victory in first of three dota 2 matches against openai. https://www.theverge.com/2018/8/23/17772376/openaidota-2-pain-game-human-victory-ai (Aug 23, 2018).
14. Simonite, T. 2018. Pro gamers fend off elon musk-backed ai bots—for now. https://www.wired.com/story/pro-gamers-fend-off-elonmusks-ai-bots/ (Aug 23, 2018).
15. Nascimento Silva, V., and Chaimowicz, L. 2015. On the development of intelligent agents for moba games. In Computer Games and Digital Entertainment (SBGames), 2015 14th Brazilian Symposium on, 142–151. IEEE.
16. Hagelbäck, J., and Johansson, S. J. 2008. The rise of potential fields in real time strategy bots. In Fourth Artificial Intelligence and Interactive Digital Entertainment Conference. Stanford University
17. Ontanón, S., and Buro, M. 2015. Adversarial hierarchical-task network planning for complex real-time games. In Twenty-Fourth International Joint Conference on Artificial Intelligence.
18. B. W. Ballard. The *-minimax search procedure for trees containing chance nodes. Artificial Intelligence, 21(3):327–350, 1983.
19. Branislav Bošanský, Viliam Lisý, Marc Lanctot, Jirí Čermák, and Mark H.M. Winands. Algorithms for computing strategies in two-player simultaneous move games. Artificial Intelligence, 237:1–40, 2016.
20. Kevin Waugh, Dustin Morrill, J. Andrew Bagnell, and Michael Bowling. Solving games with functional regret estimation. In Proceedings of the AAAI Conference on Artificial Intelligence, 2015. https://arxiv.org/abs/1411.7974.
21. Noam Brown and Tuomas Sandholm. Superhuman ai for multiplayer poker. Science, 11, 2019.