IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

SPECIAL SECTION ON ARTIFICIAL INTELLIGENCE FOR PHYSICAL-LAYER WIRELESS COMMUNICATIONS

Received August 16, 2019, accepted August 26, 2019, date of publication August 29, 2019, date of current version September 11, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2938240

A Reinforcement Learning Model Based on Temporal Difference Algorithm

XIALI LI[1], ZHENGYU LV[1], SONG WANG[1], ZHI WEI[2], AND LICHENG WU[1]
[1]School of Information Engineering, Minzu University of China, Beijing 100081, China
[2]Department of Computer Science, College of Computing Sciences, New Jersey Institute of Technology, Newark, NJ 07102, USA

Corresponding author: Licheng Wu (wulicheng@tsinghua.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61602539, Grant 61873291, and Grant 61773416, and in part by the Minzu University of China (MUC) 111 Project.

ABSTRACT In some sense, computer game can be used as a test bed of artificial intelligence to develop intelligent algorithms. The paper proposed a kind of intelligent method: a reinforcement learning model based on temporal difference (TD) algorithm. And then the method is used to improve the playing power of the computer game of a special kind of chess. JIU chess, also called Tibetan Go chess, is mainly played in places where Tibetan tribes gather. Its play process is divided two sequential stages: preparation and battle. The layout at preparation is vital for the successive battle, even for the final winning. Studies on Tibetan JIU chess have focused on Bayesian network based pattern extraction and chess shape based strategy, which do not perform well. To address the low chess power of JIU chess from the view of artificial intelligence, we developed a reinforcement learning model based on temporal difference (TD) algorithm for the preparation stage of JIU. First, the search range was limited within a 6×6 area at the center of the chessboard, and the TD learning architecture was combined with chess shapes to construct an intelligent environmental feedback system. Second, optimal state transition strategies were obtained by self-play. In addition, the results of the reinforcement learning model were output as SGF files, which act as a pattern library for the battle stage. The experimental results demonstrate that this reinforcement learning model can effectively improve the playing strength of JIU program and outperform the other methods.

INDEX TERMS Artificial intelligence, reinforcement learning, temporal difference algorithm, JIU chess.

I. INTRODUCTION

Deep learning and reinforcement learning are current popular ways of artificial intelligence [1]–[3]. Computer games research stands out as one of the notable landmarks in the progress of artificial intelligence. The study of computer game is as old as computer science itself. Charles Babbage, Alan Turing, Claude Shannon, and John Von Neumann devised hardware, algorithms, and theory to analyze and play the game of chess [4]–[6]. The notable studied games include Go, Chess, Shougi, Multiplayer poker, and so on [4]–[10]. In 1997, IBM's Deep Blue accomplished the astounding defeat of Kasparov in international chess competition [6]. From 2016, Google's AlphaGo [7], AlphaGo Zero [8], and AlphaZero for Go, Chess, and Shougi have achieved superhuman performance by using deep reinforcement learning(DRL) combined with Monte Carlo Tree Search(MCTS) [4]. From 2017, artificial intelligence god of gamblers, Libratus [9] and Pluribus [10], have defeated professional players of Texas Hold'em poker. Besides these popular games, some minority games played in some special area around the world, such as Backgammon, have been studied from the artificial intelligence view and achieved master-level play [11]. JIU chess is also one of the minority games. Compared to the above games, JIU chess research is in its infancy. The research on the computer game version of JIU chess with high performance is under-explored, if any.

Jiu chess game is a variant of traditional Tibetan chess. It is mainly played in places where Tibetan tribes gathering, such as the Ngawa Tibetan and Qiang Autonomous Prefecture, Garzê Tibetan Autonomous Prefecture, and Gannan Tibetan Autonomous Prefecture [12].

The JIU game process is divided into two sequential stages: preparation and battle. The public JIU board size is 14×14 [13]. The rules of Jiu chess are described in the following in detail:

1) Jiu chess is a 2-player-game while White side uses white stones and Black side uses black stones. Players take alternate turn.

The associate editor coordinating the review of this article and approving it for publication was Guan Gui.

2) Stones must be put on or move to the empty points on game board.
3) Jiu game starts with the empty board. The task in the preparation stage is to place one stone on a point at each move until there is no empty point on board. The task in the battle stage is to move or capture stones until one player wins the game. The movements and capturing methods are similar to those of international checkers.
4) In the preparation stage, White side plays first. Then Black plays. The 1st move and 2nd move must be placed on one of the points of the diagonal line of the central grid. Then each side begins to put one stone on one point until no empty points on the board alternatively.
5) In the battle stage, there are four actions in each turn to select:
    a) Jumping capture: while opponent's stone is adjacent to a player's stone and an empty point is direct behind it, Jumping capture can be selected. This action can be continued while a player cannot capture stones or a player end turn.
    b) Square Capture: if a player constructs a square with four of their stones nearby,he can capture its opponent's one stone located at any point on the board. In one turn, how many the square a player construct, he will can take how many its opponent's stones.
    c) Move: Move a stone to an adjacent empty point.
    d) Dalian: the square is the basic shape, and it can evolve into the Dalian, which is the most important JIU shape. There are two different Dalian shapes. This shape comprises seven same color stones and one empty point. The stone adjacent to the empty point is called the ''vital stone''. By moving the vital stone into the empty point to construct a square, they can capture one of the opponent's stone located at any point. A player with the Dalian shape can capture the opponent's stones by repeatedly moving the vital stone [13].
6) Winner: a player is declared the winner when they have constructed a fixed shape, such as the Dalian, and the opponent cannot construct any square shape or they have captured fewer than 14 stones.

Previous Studies on JIU are focused on expert knowledge and chess shapes. Chess power of all existing JIU playing engines based on those studies is low. [13]–[15]. Therefore, to improve the computer game power of JIU chess, we propose a temporal-difference algorithm-based reinforcement learning model for the preparation stage of JIU chess. This model fully take the characteristics of JIU chess into consideration. The search range of the model is limited to a 6×6 area in the middle of the chessboard according to the characteristics of JIU chess stones layouts.TD learning architecture was combined with chess shapes to construct an intelligent environmental feedback system. TD based search engine uses a forward-search strategy that the current state is updated according to the immediate reward returned by the current environment.The engine's strategies are optimized via sampling and iterative learning. The state transition values of the engine are updated during the self-play process of the learning model.The state transitions produced by the TD algorithm are output as a SGF file and used as opening libraries for the subsequent battle stage.The contribution of this study is summarized as follows.

- It is the first time that Temporal Difference (TD) algorithm is used in Jiu Chess, although it has been used in many other games like Go, Chess, checkers, and etc [11].
- The study verifies that Sarsa algorithm is more suitable to the special rules of Jiu Chess than Q-learning algorithm does.
- The TD algorithm based reinforcement learning model search a better policy to choose move than the Bayesian Network based expert system and defeat beginner by 67:33 (beginner played black).

The remainder of this paper is organized as follows. Section 2 introduces the related works. Section 3 presents the proposed model. Experimental and analysis results are presented in Section 4. Finally, Section 5 concludes the paper and briefly discusses future work.

II. RELATED WORKS

There are few Studies on JIU chess computer games [13]. In [15], classic chess shapes were designed for JIU chess using expert knowledge. Each chess shape was weighted according to the JIU game play. These stones of shape then were used to design offensive and defensive strategies for the preparation and battle stages of JIU chess. However, their method completely relies hand-crafted on expert knowledge. Reference [14] proposed to use a Bayesian network learning algorithm for the sample size of JIU game records is small. The chess shapes were extracted via statistical analysis of the existing game records. Those records were used to train Bayesian network for state transition probability of each shape. The classsic JIU shapes were used as the knowledge nodes of Bayesian network. Limited by localization of the shape transition probabilities, this method cannot predict more moves.

Quite a few reinforcement learning algorithms have been used in game research, including the dynamic planning (DP) algorithm [16], Monte Carlo tree search (MCTS) [7], [8], [17], and the temporal difference (TD) algorithm [18]. We summarize the features of these algorithms Table 1 and refer the reader to for their basic rules.

JIU chess' action space is huge and the action spaces in the preparation stage and battle stage are different. Limited by existing experimental conditions, to find the optimal solution by traversing all states is impossible. DP algorithm isn't suitable for searching in JIU chess due to the close relationship of every turn in the search path [16].

MCTS can not be used in Jiu chess game directly because of the unique two sequential preparation and battle stages

TABLE 1. Comparison of current reinforcement learning algorithms used in game research.

Algorithm	Has a model	State transition probability is known	Reward function is known	State value function	Appropriate action sequence length
DP	Yes	Yes	Yes	Traverse all states; bootstrap algorithm	Short
MCTS	No	No	No	The state value function is estimated by averaging experimentally accumulated experience;the empirical average only appears at the end of each experiment.	Long
TD	Yes	No	No	The state value function is estimated by averaging experimentally accumulated experience,and the state value function of the next instant is used to correct the state value estimate of the current instant; bootstrap algorithm.	Long

playing rule of JIU chess. The state-action of every turns in one game often have different action sequence for the special rules in Jiu chess.

TD algorithm combines the strengths of DP and MCTS. TD algorithm has better performance and easy to be applied on other popular board game [19]. There are successful examples of TD algorithm playing good in games [11]. Therefore, the TD algorithm is more appropriate for developing JIU chess search engine.

III. TD ALGORITHM BASED REINFORCEMENT LEARNING MODLE FOR JIU CHESS

1) SEARCH RANGE OF THE MODEL

The goal of the preparation stage is to build an advantageous shape for the battle stage and disrupt the shape designed by the opponent. The most basic chess shape for obtaining victory in the battle stage is the square (gate), and the Dalian shape is the derivatives of this shape. In moderate- and high-level matches, the Dalian shape is key to securing victory. The chance to win in JIU chess is mainly determined by the potentially advantageous chess shapes designed during the preparation stage. The quality of a chess shape is determined by the 6×6 central region of the chessboard. Therefore, this 6×6 area (Figure 1) is defined as the search range of the reinforcement learning model.

2) TD ALGORITHM-BASED JIU ENGINE FOR THE PREPARATION STAGE

In reinforcement learning, an individual undergoes many actions to achieve the stated goal [20], [21]. The TD algorithm is one of the most basic forms of reinforcement learning.

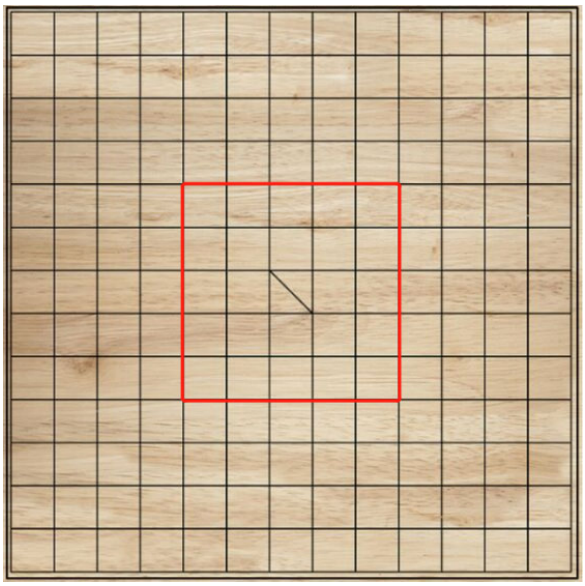

FIGURE 1. Search range of the reinforcement learning model.

It estimates and updates the value of the current state using the values of its adjacent states and a reward function [22]. TD is therefore a model-free algorithm that effectively combines the strengths of the MCTS and DP reinforcement learning algorithms. During TD reinforcement learning, an individual is placed in an interactive environment where each action generates a new state. The environment responds by returning a reward value, which depends on the innate reward mechanisms of the environment. Similar to other reinforcement learning schemes, the goal of the TD algorithm is to obtain a strategy that maximizes the cumulative reward by making continuous adjustments to the strategy; these adjustments are based on the reward values returned by the environment [23]. However, unlike other reinforcement learning schemes, the TD algorithm uses the value of the current action and its immediately adjacent states to estimate and update the value of the current state in a sampling-learning process. Hence, the current model is updated immediately after a sample is obtained. This iterative process continues until the model converges.

The JIU chess engine is considered as as a decision maker (a decision maker referred to a player), and the state of the chessboard is the interactive environment. Since the state of the chessboard changes as the player lay their stones, the interaction between the player and the game environment is mediated by laying of the stones. The game environment then returns a reward value according to the player's behavior and the state transitions of the chessboard. The construction of an internal reward mechanism is therefore the core problem at hand. After the JIU chess engine obtains feedback through this reward mechanism, the value of the current state is then updated according to the feedback of its immediately adjacent states. The engine then derives optimal action sequences from this learning process, and increases its chess-playing strength [24].

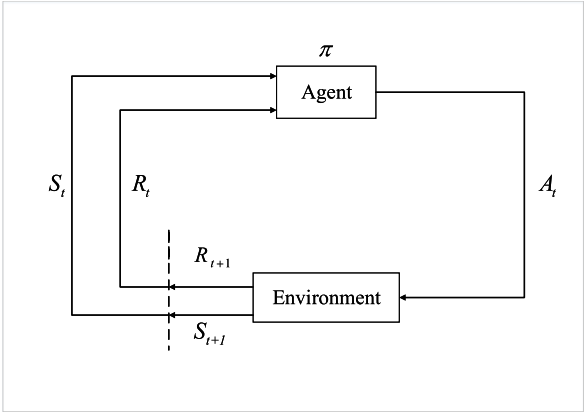

FIGURE 2. Interactions between the agent and the environment.

3) SYSTEM ENVIRONMENT OF THE GAME MODEL

In TD learning, the self-play process of the JIU chess engine is referred to as the agent. The agent interacts with the system environment, and the feedback from these interactions is used to iteratively update the strategies of the JIU chess engine [25]. The system environment refers to the JIU chessboard, which provides the foundation for this learning process. The system environment must provide feedback according to the agent's actions and changes in the chessboard's state. The accuracy of this feedback ultimately determines the outcome of the TD learning process. For example, in each iteration, the state initializing function specifies that the first two stones must be placed at the ends of the central diagonal line. After the agent has taken some action, the system environment provides an estimated reward value based on the environment's sensing mechanisms. In our TD learning model, a two-dimensional (2D) Q table is used to store the value functions with its entries (board state, action space).

In this work, board state and action space of JIU chess are expressed as follows. Board state: A variable-length tuple $\mathbf{S} = (a_1, a_2, a_3 \ldots a_n)$ is used to represent the board state, where a_1 to a_n represent the action sequence that leads to the JIU chessboard's current state, and each element is temporally linked. In other words, the current board state is formed by a time series of actions. Action space: a_i represents an action that leads to the current board state. Since each stone has two attributes, color and coordinates, the actions in the preparation stage are expressed as (stone, x, y). The "stone" describes the player who made the action, while (x, y) represents the coordinates of the action.

4) AGENT-ENVIRONMENT INTERACTIONS IN JIU CHESS

The interactions between the agent and the system environment are illustrated in Figure 2 [26].

The "Environment" is the game system of JIU chess, i.e., the JIU chessboard. As the agent interacts with the environment, the latter provides feedback about the former's actions and the state actions based on the rationality of these actions. Hence, the environment is the foundation for TD-learning by the agent.

(1) The agent represents the two sides that are playing against each other in the self-play process of the JIU chess engine. Given a current board state of $\mathbf{S} = (a_1, a_2, a_3 \ldots a_t)$, the highest valued $(\mathrm{S}, A_\mathrm{t})$ entry in the Q table is selected because At is the most valuable action in the current board state.

(2) The environment immediately returns a reward value according to the action taken by the agent (the reward mechanism is described in Section "The effect of an action on the time-action sequence"). The multi-dimensional reward mechanism returns values immediately, which is one of the parts that make up the reward target, R_t. The constitution of the reward target is different in each TD algorithm, which is Sarsa and Q-learning, but each TD algorithm always contain an immediate reward from the environment. The reward targets of the state-action-reward-state-action (Sarsa) algorithm and the Q-learning algorithm are shown in Equations (1) and (2), respectively.r_t is the Val returned by the multi-dimensional reward mechanism.

$$R_\mathrm{t} = \mathrm{r_t} + \varepsilon \mathrm{Q}\left(\mathrm{s_{t+1}}, \mathrm{a_{t+1}}\right) \tag{1}$$

$$R_\mathrm{t} = \mathrm{r_t} + \varepsilon \max_{\mathrm{a_{t+1}}} \mathrm{Q}\left(\mathrm{s_{t+1}}, \mathrm{a_{t+1}}\right) \tag{2}$$

(3) Based on the preparation-stage action that is being taken, the JIU chess environment returns a predicted reward value (R_t), the next state (S_{t+1}), and the reward target (R_{t+1}). The agent is then updated according to the difference between the R_{t+1} and R_t values returned by the environment. This updating process is crucial for TD-learning by the agent, and this is where the TD reinforcement learning algorithm differs from other reinforcement learning algorithms [27].

In the TD-learning process, the JIU chess agent obtains feedback through interactions with its environment and by sensing its environment; this feedback is then used to update the agent. In the JIU chess TD model, the agent interacts with its environment via preparation-stage actions. The selection of preparation-stage actions involves two of the most important concepts in reinforcement learning: exploration and exploitation. Exploration refers to the selection of actions that have never been taken before to explore a greater range of possibilities. Exploitation is the selection of the previously taken actions according to the current situation, thus refining the actions that are currently known [28]. In this work, the action selection strategy of the agent is determined by ϵ_greedy $\in$ (0,1). If ϵ_greedy is 0.9, there is a 90% probability that the agent will select the action with the highest estimated value in the current strategy and a 10% probability that the agent will select a random action. This helps to prevent the algorithm from falling into a local optimum.

5) UPDATING STRATEGY

The Sarsa and Q-learning TD algorithms were used in the updating process. Sarsa is an on-policy TD-learning algorithm. In Sarsa, the state-action pair's value function,Q_π(s, a), which indicates the reward of all actions (a) that are possible in the current state (s), is iteratively updated under

strategy π [29]. The updating process of the action value function in Sarsa is described in Equation (3):

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha(R_t + \varepsilon Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)) \quad (3)$$

Equation (3) shows that each update of the Sarsa algorithm is related to the $(S_t, a_t, s_{t+1}, a_{t+1})$ terms of JIU chess. S_t is the current board state, a_t is the preparation-stage action that was performed in state S_t, S_{t+1} is the next board state of the (S_t, a_t) state-action pair, and a_{t+1} is the preparation-stage action that will be performed in S_{t+1} under the current strategy. The Sarsa algorithm will select the a_{t+1} action in the next state, s_{t+1}, when laying the next stone.Q-learning is an off-policy TD-learning algorithm. Unlike Sarsa, where the same strategy is used for action selection and for updating the action value function, the $\max_{a_{t+1}} Q(s_{t+1}, a_{t+1})$ term in Q-learning is only used to update the Q-table [30], and the a_{t+1} action may not be selected in the next state. The updating of the action value function in Q-learning is described in Equation (4). The Q table is updated according to the difference between the predicted value at time t and the actual one-step reward plus the predicted longer-term value after that one-step reward given by the JIU environment's multi-dimensional feedback mechanism.

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha(R_t + \varepsilon \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)) \quad (4)$$

Both Sarsa and Q-learning algorithms use the same TD reinforcement learning framework but differ in their updating strategies. Therefore, the TD-learning environment and the interactions between the JIU chess agent and the game environment were separated into a different package, while the Sarsa and Q-learning algorithms were used as updating strategies for the TD algorithm.

6) MULTI-DIMENSIONAL SENSING AND FEEDBACK MECHANISM

The heart of the TD algorithm is the provision of feedback for each action and the state transition made by the agent. In this work, the environmental factors that significantly affect the game situation and agent decisions were chosen as feedback factors. These factors return the reward values according to the actions and state transitions chosen by the agent. The chosen factors are the spatial values of the current action (default positions and controllable area) and the effects of the time-action series (the advantageous shape formed by the action sequence). At present, there are no comprehensive shape and pattern libraries available for the static evaluation of game situations in JIU chess. Therefore, our multi-dimensional sensing-feedback mechanism uses a two-dimensional three-aspect assessment metric that was designed using expert knowledge and the currently available literature on JIU chess. The value returned by the feedback mechanism is calculated using Equation (5).

$$Val = e_1 + e_2 + e_3 \quad (5)$$

TABLE 2. Default position values of JIU chess stones in the spatial dimension.

1	1	1	1	1	1
1	50	50	50	50	1
1	50	10^6	10^5	50	1
1	50	10^5	10^6	50	1
1	50	50	50	50	1
1	1	1	1	1	1

where e_1 is the value of the default position of the current action space, e_2 is the area expected to be controlled by the played stones, and e_3 is the expected influence of the current stone in the time-action series. Val is the expected reward returned by the multi-dimensional sensing-feedback mechanism after an action is taken.

a: DEFAULT POSITION VALUES

A game of JIU chess can be divided into two stages: preparation and battle. In the preparation stage, the first step is to place two stones at the ends of the diagonal lines in the middle of the board. The removal of these stones marks the beginning of the battle stage. Since the first actions of each stage are taken around the center of the board (as represented by the two central stones), this is the area where the stones are initially laid out during the preparation stage. Therefore, the central locations of the board are more valuable than the board's peripheral positions. Default values were assigned to the positions in the spatial dimension of the chess board. The central diagonal has the highest value at 106 while the values of the other positions decrease radially. The values of the 36 positions that make up the central region are shown in Table 2 [13]; all other positions have a default value of 0.

b: CONTROLLABLE AREA

The rules of JIU chess are simple; the goal of the game is to capture the opponent's stones via "jumping capture" or "square capture," or Dalian shapes to disrupt the shapes that could threaten one's stones. Therefore, the value of each chess stone in the spatial dimension also depends on the stone's influence on its surroundings, i.e., the ability of a stone to capture other stones or form shapes by moving within its available range of motion. Hence, the area that can be controlled by a stone is also an important factor in evaluating an action.

At the beginning of the battle stage, only a few stones in the central region are able to move. As these stones are the only stones that can affect their surroundings (and the entire game), they have the highest value among all stones. Towards the end of the battle stage, when one of the players has no more than 14 stones, all the stones belonging to that player can move to any position on the chess board and thus affect the overall state of the game through their movements. Therefore, if a player cannot form some stone shapes before the weaker side has fewer than 14 remaining stones, the weaker side will win. In this work, the area that can be controlled by each stone was analyzed according to the

TABLE 3. Spatial dimension: Assignment of values according to the area that can be controlled by a stone.

Move to an adjacent position	Able to make a single jumping capture	Able to make n consecutive jumping captures	Able to construct 'Dalian'
1	5	5*n	10^4

TABLE 4. Valuation of the chess shapes that can be formed by time-action sequences in a game of JIU chess.

Chain	Triangle	Dalian
2	2	10^4

JIU rules to enable the assignment of values. The JIU chess stones were classified into four types by the area they control: stones that can only move by one step, stones that can perform jumping capture, stones that can make consecutive jumping captures, and stones that can make Dalian. The value of each class is shown in Table 3.

c: THE EFFECT OF AN ACTION ON THE TIME-ACTION SEQUENCE

The capture of an opponent's stone through an action is one of the consequences of JIU chess. However, the shape of certain chess shapes such as the Dalian shape will greatly increase the win probability because the Dalian shape is one of the basic shapes required to win a game of JIU chess. In high-level games, the shape of a Dalian shape is equivalent to victory. If a player succeeds in forming powerful chess shapes such as the Double Dalian, that player is considered to have won an overwhelming victory. Therefore, the chess shape by an action sequence is the most important aspect of situation assessment in a JIU chess game. The value of a turn significantly increases if a stone movement or capture in that turn leads to a chess shape. In this work, a set of chess shapes were formulated using the rules of JIU chess and expert knowledge, and the values were assigned to these shapes. The valuation of the chess shapes is displayed in Table 4.

IV. EXPERIMENTAL RESULTS AND ANALYSIS

The results of the self-play by our reinforcement learning model for JIU chess were outputted as SGF files, which were then used to generate an opening library. This opening library provides a library of actions for the preparation-stage engine; a reinforcement learning JIU chess engine was thus formed by combining this preparation-stage engine with the battle-stage engine described in [14]. To examine the performance of Sarsa and Q-learning updating algorithm in our reinforcement learning model, the JIU chess uses a Bayesian network-guided shape transitioning strategy.

The actions taken by the preparation-stage engine were learned through a Bayesian network, whereas the offensive and defensive strategies taken by the battle engine were selected on the basis of expert knowledge-based evaluations.

TABLE 5. Configuration of the test environment.

Computer model	CPU	GPU	OS
TF T500	i7-8750H	GTX1050Ti	Windows10

TABLE 6. Configuration of the tested JIU chess engines.

Learning algorithm	Preparation stage	Battle stage
Sarsa	Sarsa opening library	Static evaluation
Q-learning	Q-learning opening library	Static evaluation
Bayesian network	Bayesian network	Static evaluation

The human player that played against the JIU chess engine is a medium-level JIU chess player named Naota, who works in the Computer Games Laboratory of the Minzu University of China and won a JIU chess tournament in Minzu University.

A. CONFIGURATION OF THE TEST ENVIRONMENT AND INTRODUCTION OF DIFFERENT JIU ENGINES

A uniform testing environment, whose hardware components are shown in Table 5, was set up to eliminate the effects of environmental factors on the results of our tests. All versions of our JIU chess engines were tested in the same test environment.

Our JIU chess engine has two different versions; each version uses a different updating strategy in the TD algorithm (Sarsa or Q-learning). Both versions of our JIU chess engine were played against the Bayesian network-based JIU chess engine, and the learning outcome of the adopted TD algorithm was compared via the win rates. The configurations of the JIU chess engines are presented in Table 6.

B. TESTS ON DIFFERENT UPDATING ALGORITHMS

The JIU chess engines were tested using the TD algorithm opening libraries with different numbers of iterations. To minimize the impact of other external factors, the ϵ_greedy parameter was set to 0.9 in all TD algorithms. The win rates of the Sarsa JIU chess engine (i.e., the program using the Sarsa algorithm to generate its opening library) against the Bayesian network-based JIU chess engine (i.e., the program using the Bayesian network-guided shape transition strategy to generate its opening library) are displayed in Figure 3. The horizontal axis indicates the number of learning iterations used by the Sarsa JIU chess engine, whereas the vertical axis indicates the win rate of the Sarsa JIU chess engine against the Bayesian network-based JIU chess engine. The Sarsa JIU chess engine converges after 80,000 iterations (games played), and its win rate stabilizes around 68%. The win rate of the Q-learning JIU chess engine against the Bayesian network-based JIU chess engine is illustrated in Figure 3. The Q-learning JIU chess engine converges after 40,000 iterations, and its win rate stabilizes around 66%. The Sarsa algorithm converges more slowly during the reinforcement

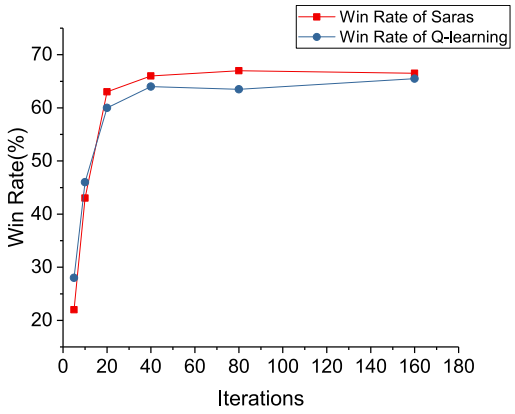

FIGURE 3. Win rate of the JIU chess engine using the Sarsa and Q-learning opening libraries.

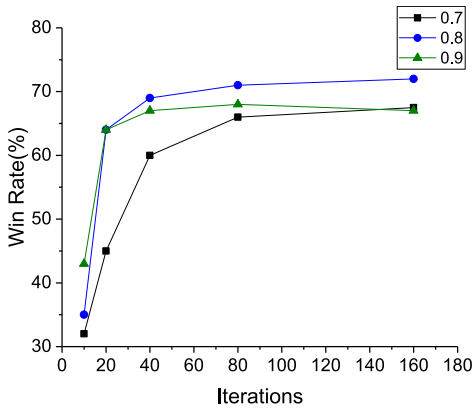

FIGURE 4. Win rates of the Sarsa JIU chess engine with different ϵ_greedy values.

learning process but ultimately produces better results than Q-learning algorithms.

C. EFFECT OF PARAMETERS ON SARSA ALGORITHM

The following tests were performed using the Sarsa algorithm only, with ϵ_greedy and the number of training iterations used as independent variables. The effects of ϵ_greedy on the convergence of the Sarsa algorithm were determined by evaluating the win rates of the Sarsa JIU chess engine against the Bayesian network-based JIU chess engine that changed with the value of ϵ_greedy. Lower ϵ_greedy values allow the JIU chess engine to take random actions more frequently during TD learning and thus avoid falling into local optima. Lower ϵ_greedy values therefore result in higher win rates, if provided a sufficient number of training iterations. However, this comes at the cost of a slower rate of convergence. An ϵ_greedy value of 0.8 produces an acceptable rate of convergence, and the win rate of this Sarsa JIU chess engine reaches its maximum after 160,000 rounds of iteration (Figure 4). Battle stage is closely related to preparation stage in JIU chess, so the search path is much longer than other games. Thus updating value in Q table by Q-learning is inefficient compared to Sarsa method due to the long game search path.

TABLE 7. Wins and losses against Song Wang (junior level human JIU chess player).

Player	Wins as White	Wins as Black
Our JIU chess engine	61	67
Song Wang	39	33

TABLE 8. Wins and losses against Naota (medium-level human JIU chess player).

Player	Wins as White	Wins as Black
Our JIU chess engine	32	36
Naota	68	64

D. TESTS AGAINST HUMAN PLAYERS

The results of our previous tests show that the Sarsa algorithm with an ϵ_greedy value of 0.8 provides the best learning outcomes. Using this combination, 160,000 rounds of training were performed, and the resulting JIU chess engine was played against human opponents (including junior and medium chess players).

The program and the junior JIU chess player, Song Wang, played 200 games. The program and Song Wang held the white in turns. The winning results of the 200 games are shown in Table 7. The winning rate of the JIU chess program is 0.61 as the white and 0.67 as the black. This demonstrates that the JIU chess program using the TD algorithm can outperform a junior chess player.

The program and the medium-level JIU chess player, Naota, also played 200 games and held the White in turns. The winning results are shown in Table 8. Our reinforcement learning JIU chess engine has a win rate of 0.32 as the white and 0.36 as the black. Although our engine showed a weaker performance than the medium-level human player, it performed better than the chess shape-based JIU chess engine designed in [14], whose win rate against Naota is less than 0.3.

V. CONCLUSION AND FUTURE WORK

Based on the JIU chess rules, we proposed to employ TD algorithms for reinforcement learning in JIU chess engines and constructed a reinforcement learning model for the preparation stage of JIU chess. The Sarsa and Q-learning algorithms were used to train the reinforcement learning model, and the learning outcome was output as SGF file to generate opening libraries for the battle stage. The incorporation of the TD algorithm in the JIU chess engine resulted in an agent that has the ability to learn, and the playing strength of the JIU chess engine was significantly improved by the TD algorithm. However, due to the lack of systematic expert knowledge, the sensing-feedback mechanism of the TD algorithm is only a "best effort" attempt in ensuring that the learning outcomes are indeed rational. Furthermore, the iteratively updated values in TD-based reinforcement learning are intrinsically biased because it is very difficult to perform

bias-less estimations using the TD algorithm. In future work, we plan to study objective function-based learning via the incorporation of a neural network model in our JIU chess engines and use TD learning to optimize the parameters of the neural network. It is expected that the guidance provided by the neural network will further improve the learning outcomes of JIU chess engines.

ACKNOWLEDGMENT

(Zhengyu Lv and Song Wang contributed equally to this work.) They programmed the JIU chess software by3 cooperation.

REFERENCES

[1] G. Gui, H. Huang, Y. Song, and H. Sari, "Deep learning for an effective nonorthogonal multiple access scheme," *IEEE Trans. Veh. Technol.*, vol. 67, no. 9, pp. 8440–8450, Sep. 2018.

[2] Y. Wang, M. Liu, J. Yang, and G. Gui, "Data-driven deep learning for automatic modulation recognition in cognitive radios," *IEEE Trans. Veh. Technol.*, vol. 68, no. 4, pp. 4074–4077, Apr. 2019.

[3] J. Sun, W. Shi, Z. Yang, J. Yang, and G. Gui, "Behavioral modeling and linearization of wideband RF power amplifiers using BiLSTM networks for 5G wireless systems," *IEEE Trans. Veh. Technol.*, to be published.

[4] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, and T. A. Lillicrap, "A general reinforcement learning algorithm that masters chess, shogi, and go through self-play," *Science*, vol. 362, no. 6419, pp. 1140–1144, Dec. 2018.

[5] F. H. Hsu, *Behind Deep Blue: Building the Computer that Defeated the World Chess Champion*. Princeton, NJ, USA: Princeton Univ. Press, 2004.

[6] M. Campbell, A. J. Hoane, Jr., and F.-H. Hsu, "Deep blue," *Artif. Intell.*, vol. 134, nos. 1–2, pp. 57–83, Jan. 2002.

[7] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, and S. Dieleman, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, p. 484, Jan. 2016.

[8] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, and Y. Chen, "Mastering the game of go without human knowledge," *Nature*, vol. 550, no. 7676, p. 354, Oct. 2017.

[9] N. Brown and T. Sandholm, "Safe and nested subgame solving for imperfect-information games," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 689–699.

[10] B. Noam and S. Tuomas, "Superhuman AI for heads-up no-limit poker: Libratus beats top professionals," *Science*, vol. 359, no. 6374, pp. 418–424, 2018.

[11] G. Tesauro, "TD-Gammon, a self-teaching backgammon program, achieves master-level play," *Neural Comput.*, vol. 6, no. 2, pp. 215–219, 1994.

[12] P. Shotwell and D. SuodaChucha, "The research of tibetan chess," *J. Tibet Univ.*, vol. 9, no. 2, pp. 5–10, 1994.

[13] X. Li, S. Deng, X. Wang, Y. Li, L. Wu, and J. Duan, "Review of research on computer games for tibetan chess," in *Proc. 14th Int. Conf. Dependable, Autonomic Secure Comput.*, Aug. 2016, pp. 97–99.

[14] S. T. Deng, "Design and implementation of jiu game prototype system and multimedia courseware," Ph.D Dissertation, Dept. Sci. Technol., Minzu Univ. China, Beijing, China, 2017.

[15] X. Li, S. Wang, Z. Lv, Y. Li, and L. Wu, "Strategy research based on chess shapes for tibetan JIU computer game," *ICGA J.*, vol. 40, pp. 1–11, 2018.

[16] J. M. Vaccaro, C. C. Guest, and D. O. Ross, "Evolutionary programming for goal-driven dynamic planning," *Appl. Sci. Comput. Intell.*, vol. 4739, pp. 28–40, Mar. 2002.

[17] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, and S. Petersen, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, p. 529, 2015.

[18] L. N. Niu, X. C. Zhang, and Y. Tang, "9*9 go machine game system with time difference algorithm," *CAAI Trans. Intell. Syst.*, vol. 7, no. 3, pp. 278–282, 2012.

[19] J. Schaeffer, M. Hlynka, and V. Jussila, "Temporal difference learning applied to a high-performance game-playing program," in *Proc. 17th Int. Joint Conf. Artif. Intell. Vol.* San Mateo, CA, USA: Morgan Kaufmann, 2001, pp. 529–534.

[20] D. Shawky and A. Badawi, "A reinforcement learning-based adaptive learning system," in *Proc. Int. Conf. Adv. Mach. Learn. Technol. Appl.*, Jan. 2018, pp. 221–231.

[21] H. Huang, J. Yang, H. Huang, Y. Song, and G. Gui, "Deep learning for super-resolution channel estimation and doa estimation based massive MIMO system," *IEEE Trans. Veh. Technol.*, vol. 67, no. 9, pp. 8549–8560, Sep. 2018.

[22] J. Y.-H. Ng and L. S. Davis, "Temporal difference networks for video action recognition," in *Proc. IEEE Winter Conf. Appl. Comput. Vis. (WACV)*, Mar. 2018, pp. 1587–1596.

[23] M. McPartland and M. Gallagher, "Reinforcement learning in first person shooter games," *IEEE Trans. Comput. Intell. AI Games*, vol. 3, no. 1, pp. 43–56, Mar. 2010.

[24] H. Cuayáhuitl, "Deep reinforcement learning for conversational robots playing games," in *Proc. IEEE-RAS 17th Int. Conf. Humanoid Robot. (Humanoids)*, Nov. 2017, pp. 771–776.

[25] F. G. Glavin and M. G. Madden, "DRE-Bot: A hierarchical first person shooter bot using multiple Sarsa(λ) reinforcement learners," in *Proc. 17th Int. Conf. Comput. Games (CGAMES)*, Aug. 2012, pp. 148–152.

[26] F. Zhu, H. Zhu, Y. Fu, D. Chen, and X. Zhou, "A kernel based true online sarsa (λ) for continuous space control problems," *Comput. Sci. Inf. Syst.*, vol. 14, no. 3, pp. 1–14, 2017.

[27] M. A. Pengwei and D. Pan, "Sarsa(λ) algorithm based on heuristic function," *Comput. Digit. Eng.*, vol. 44, no. 5, pp. 825–828, 2016.

[28] M. Tokic and G. Palm, "Value-difference based exploration: Adaptive control between epsilon-greedy and softmax," in *Proc. Annu. Conf. Artif. Intell.*, 2011, pp. 335–346.

[29] S.-L. Chen and Y.-M. Wei, "Least-squares sarsa(λ) algorithms for reinforcement learning," in *Proc. 4th Int. Conf. Natural Comput.*, vol. 2, Aug. 2008, pp. 632–636.

[30] X. Guo, S. Singh, H. Lee, R. L. Lewis, and X. Wang, "Deep learning for real-time atari game play using offline monte-carlo tree search planning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 3338–3346.

XIALI LI was born in Henan, China. She received the B.S. and M.S. degrees in computer science and technology from Xi'an Jiaotong Univerisity (XJTU), Xi'an, China, in 2001 and 2004, respectively. From October 2008 to August 2009, she was a Visiting Scholar with the University of Edinburgh, U.K. She is currently an Associate Professor with the Minzu University of China. She has authored or coauthored more than three books, 50 articles, and two inventions. Her research interests include computer games, machine learning, and artificial intelligence.

ZHENGYU LV was born in Guiyang, Guizhou, China, in 1992. He received the bachelor's degree in engineering from the Minzu University of China, in 2015, where he is currently pursuing the degree in modern education technology. His research interests include computer game and artificial robotics.

SONG WANG was born in Baoding, Hebei, China, in 1994. He received the bachelor's degree in engineering from Beijing Union University, in 2017. He is currently pursuing the degree in modern education technology with the Minzu University of China. His research interests include computer game and artificial robotics.

ZHI WEI received the B.S. degree from Wuhan University, China, and the Ph.D. degree from the University of Pennsylvania, in 2008. He is currently an Associate Professor with the Department of Computer Science, New Jersey Institute of Technology. His research interests include data mining, statisical modeling, and machine learning with applications to data enriched fields.

LICHENG WU was born in Jiangxi, China. He received the bachelor's degree from the Beijing University of Aeronautics and Astronautics (BUAA), Beijing, China, in 1995, and the Ph.D. degree in robotics from the Institute of Robotics, BUAA, in 2000. In 2001 and 2002, he held a postdoctoral position with the State Key Laboratory of Intelligent Technology and Systems, Department of Computer Science and Technology, Tsinghua University. He was an Associate Professor with Tsinghua University, in 2008. He moved to the School of Information Engineering, Minzu University of China, in November 2009. From November 2006 to November 2007, he was a Visiting Scholar with the Polytechnic di Milano and Cassino University, Italy. He has been a Full Professor with the Minzu University of China, Beijing, China, since January 2013, where he is currently a Full Professor and the Dean with the School of Information Engineering. He has authored three books, more than 80 articles, and more than three inventions. His research interests include artificial intelligence, computer games, and robotics.