Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 160 (2019) 241–247

Procedia Computer Science

www.elsevier.com/locate/procedia

The 10th International Conference on Emerging Ubiquitous Systems and Pervasive Networks (EUSPN 2019)
November 4-7, 2019, Coimbra, Portugal

Development of a Computer Player for Seejeh (A.K.A Seega, Siga, Kharbga) Board Game with Deep Reinforcement Learning

Ahmad Aljaafreh[a,*], Naeem Al-Oudat[a]

[a]*Tafila Technical University, Tafila, Tafila 66110, Jordan*

Abstract

Recent years have proven the existing room of deep reinforcement learning (DRL) applications. DRL has been utilized as an AI computer player in many board games. Seejeh is an ancient board game, where no one attempts to create an AI system that is able to learn to play it. Seejeh is a two-player, zero-sum, discrete, finite and deterministic game of perfect information. Seejeh board game is different from all other strategic board games. It has two stages; Positioning and moving. Player place two tiles at each action in stage one. A player might have a sequence of moves in the second stage unlike Othello and Go. In this work, we develop an automated player based solely on reinforcement learning, without human data, guidance or domain knowledge beyond game rules. This paper presents a self-play algorithm utilizing DRL and search algorithms. The system starts with a neural network that knows nothing about the game of Seejeh. It then plays games against itself, by combining this neural network with powerful search algorithms. To the best of our knowledge, we are the first who develop an agent that learns to play Seejeh game.

Keywords: Board game; deep reinforcement learning; search; MCTS; Minimax; self-play; Seejeh;

1. Introduction

The first mention of Seejeh in the literature is in 1889. Seejeh boards are found engraved on the summit of the Great Pyramid of Cheops and on fallen roofing slabs at the temple of Kurna, in Egypt. Seejeh is a game with two players, zero sum, perfect information, deterministic, discrete and sequential. Seejeh can be described as an abstract strategy combinatorial game like Go, Chess, Checker, Othello and Tic-Tac-Toe games. It is played on 5x5, 7x7 or 9x9 board, where 7x7 board is the most popular. In 7x7 board, each one of the players has a twenty-four pieces with two different colors. One player has dark pieces while the other has light ones. Seejeh is different from other board games because it has two stages; namely positioning and moving. Players alternate turns in both stages. In positioning stage,

* Corresponding author. Tel.: +962-777-411-126.
E-mail address: a.aljaafreh@ttu.edu.jo

10.1016/j.procs.2019.09.463

players take turns placing two of pieces at a time of their assigned color. Players cannot place any piece on the center of the board. After placing all the forty-eight pieces, stage two starts. In stage two, player 1, who starts placing, starts moving one of his pieces to the only empty place (board center). If the move results on a bounded piece or more of the opponent pieces then the bounded pieces are taken out. Any capture sequence may be chosen as long as all possible captures are made. Each player tries to move to catch one of his adversary's pieces between two of his own pieces in horizontal or in vertical lines, not in a diagonal. Pieces can only move one cell at a time and cannot skip cells or jump over other pieces, whether their own or opponent's. If, however, in moving, one player places his own piece between two of his opponent's, he suffers no loss; a capture must be forced by the opponent [1]. When player 1 has no legal moves, player 2 takes turn in moving. After moving a piece to any empty space, the opponent's pieces that are in a straight line and bounded by two of his pieces are taken out. A piece may only move to empty place and it may not move diagonally. In the meantime, whenever player 2 is able to offer a number of pieces that he can take out without losing his position on the board, he takes it out of the patch and asks his opponent (the first player) to take out two pieces for each one. If the pieces of player 1 get trapped so that they cannot make any move then player 2 wins the game. The pieces of player 2 may get trapped only once at the first move of player 1. In this case, player 1 has to free up cell for player 2 to move. This cell should be a neighbor of the cell of the original place of the first move.

Seejeh game is unlike other board games, it involves a much delayed reward structure. Rewards are obtained after the end of the moving stage of the game to accurately describe how well each player is performing.

In the moving stage of Seejeh game, Minimax algorithm is used to play (move) tiles. In the first stage (positioning), deep neural networks and a *tabula rasa* reinforcement learning algorithm is utilized to decide the best action to play.

The rest of the paper is organized as follows. Section 2 brings the current achievements in this domain. Seejeh game and its strategies are described in Section 3. Section4 gives a general description of our proposed method. In Section 5, we provide the conclusions.

2. Related Work

Reinforcement learning employment in learning agents is relatively new that was evolved in the late 1980s [2]. Since then, RL has employed in developing self learning agents for computer games. Agents that create strategies by self-play learning has been studied by many researchers in the field of AI [3], [4]. Go is an example of self-play learning agents, which has been studied in [5]. Authors in [6] tried a self-play methods with Chess. Backgammon as a stochastic game was also studied for playing using a self-play and knowledge based methods in [7].

Deep neural network agents were considered by David Silver et. all. in [8], [9] and [10] for Go game. Their approach in Go were generalized for Chess and Shogi [11]. Othello game were studied in [12] using an experimental approach of several CNN-based deep neural networks.

Minimax algorithm was also used for board games [13]. Othello-playing programs, that are excellent computer players use a Minimax search along with state evaluation function [14].

3. Seejeh

Seejeh game consists of two stages; The positioning stage and the moving stage. Players alternate turns in both positioning and moving. This section provides an illustrative examples of a 5x5 full Seejeh game for both stages (positioning and moving).

3.1. Positioning Stage

Fig. 1 shows a full positioning stage for Seejeh game, where black player starts the game with black tiles (tile and piece are used interchangeably through out this paper).

In action one, as shown in the upper left board in the figure, black player placed two tiles in (c2, c5) according to Seejeh rules that is explained in the Introduction. This positioning strategy enables him to capture opponent's tile in c4. This capture happens in moving stage where black player moves tile in c2 to c3, center of the board (which is left

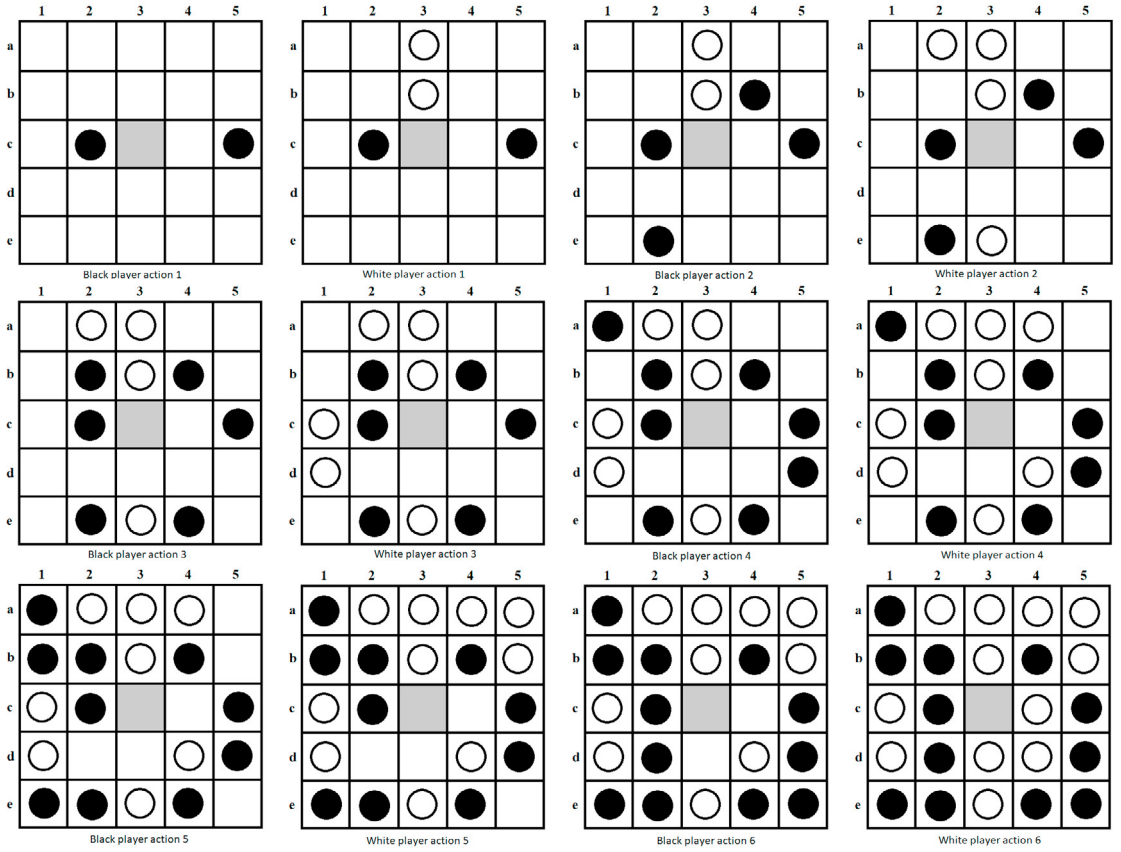

Fig. 1. Full positioning stage for Seejeh game, where black player starts the game with black tiles.

blank according to Seejeh rules). For the same reasoning, positioning in (d3, a3), (b3, e3) and (c1, c4) results in the same strategy. After black player placement of his two tiles then the turn is for white player.

White player in his action number one placed two tiles in (b3, a3). This strategy prevents black player from capturing white tile in b3. The same strategy is applied if his tiles were placed in (d3, e3).

These strategies are built to minimize the likelihood of loosing tiles in the moving stage that follows. This make Seejeh a unique strategic board game as compared with other popular board games (e.g., Go, Checkers, Othello, Chess, etc.). Players keep alternating their turns until they fill the board.

Each player in his turn plans to place his tiles while he is evaluating his choices according to the outcome of moving stage. For example in last action of black player, Fig. 1, action number six, player has six choices {(c4, d2), (c4, d3), (c4, e5), (d2, d3), (d2, e5), (d3, e5)}. Black player evaluates all choices to maximize his opportunity of wining. According to his plan the best choice is (d2, c5), to force white player to place his last two tiles in (c4, d3).

For example, black player evaluation of (d2, d3) choice is as follows; after playing the first move, which is moving the tile in c2 to the empty cell c3, black player captures the white tile in c4. Then he has to continue capturing the tile in d4. White player can move his tile in b3 to c3 capturing black tile in d3 then he has four choices {(c3 to b3), (c3 to c4), (c3 to d3), (c3 to c2)}, in each one of them he can capture one black tile.

3.2. Moving Stage

Fig. 2 shows a full moving stage (stage two) for 5x5 Seejeh game. Black player starts the moving stage as he starts the positioning stage. Black player starts by moving his black tile in c2 to the only empty cell (center of the board) which is c3. Black player has no choice other than this move. After this move, the white tile in c4 is captured since

it is bounded between the black tiles in cell c3 and c5. If the player has more than one capture sequence then he can choose any as long as all possible captures are made. Then the black player has to continue playing another action. The action is moving the tile in c3 to the empty cell c4 to capture the white tile in d4. The capture is possible since d4 is bounded between the black tiles in c4 and e4, Fig. 2 in black player action 2. Black player still has another move, which is moving his tile in c4 to d4 to capture the white tile in d3. White player takes the turn. White player has three different possible actions to take, {(b3 to c3), (c1 to c2), (e3 to d3)}. White player chose (e3 to d3) to capture the black tile in d3. Then players keep alternate turns until reaching the pre-last board configuration in Fig. 2.

Since black player starts the game, white player has a privilege of providing a number of his pieces that he can take out without losing his position in the board. This enables him to ask his opponent to take out two pieces for each one. White player is able to provide the white tiles in cells a5 and a3 without losing his position in the board. In this case, black player has to take out two of his tiles for each one of white tiles in a5 and a3. Black player has the choice to take out any of his tiles. Black player best choice is taking out tiles in e4, e5, d5 and c3. After taking out four of the black tiles for two of the white tiles, the number of white tiles equals number of black tiles as shown in last board in Fig. 2. This makes the white player wins the game, since he has the privilege of two for one.

4. Methods

Most of the automated computer player of board games trained on datasets of expert human play. Self-play learning scheme is a must in Seejeh, since there is no available dataset for the game. Seejeh self-play learning system includes two main agents; namely positioning and moving agents. The positioning player work ideas were extracted from Alphazero paper [8] and applied to the Seejeh game. Minimax is used in the second stage which is the moving stage.

4.1. Reinforcement Learning of Positioning Agent

The system starts with a neural network that knows nothing about the game of Seejeh other than the rules of the game. Then, it plays games against itself, by combining neural network with Monte Carole Tree Search (MCTS) and Minimax search algorithms. As it plays, the neural network is tuned and updated to predict actions, besides the winner of the game. This updated neural network is then recombined with the search algorithm to create a new, stronger version of the computer player, and the process repeats again. In each iteration, the performance of the system improves by a small amount, and the quality of the self-play games increases, leading to more and more accurate neural network and stronger versions the computer player.

This technique is powerful because it is no longer constrained by the limits of human knowledge. The computer player only uses the black and white pieces from the board as its input without any hand-engineered features. It uses one neural network which combines policy network to select the next move to play and a value network to predict the winner of the game from each position, allowing it to be trained and evaluated more efficiently.

4.2. Policy and Value Deep Neural Network

The deep neural network is the core of the learning algorithm. The neural network f_θ is parameterized by θ. It takes state of the board s as input. It gives two outputs: a policy $\vec{p_\theta}(s)$ which is the probability vector over all possible actions and a continuous value of the board state $v_\theta(s) \in [-1, 1]$. In training phase, the network is initialized randomly. The neural network is provided with training examples at the end of each game of self-play. These examples has the form $(s_t, \vec{\pi_t}, z_t)$. $\vec{\pi_t}$ is an improved policy estimate after MCTS starting from s_t. z_t is the final score of the game from current player perspective. The final score is obtained from the moving agent which uses Minimax to a certain depth search. The neural network parameters θ are updated to minimize the error as in the following loss function:

$$l = \sum_t (v_\theta(s_t) - z_t)^2 - \vec{\pi_t}.log(\vec{p_\theta}(s_t)) \tag{1}$$

Overtime the neural network will learn the value of the state that indicates weather the state is leading to loss or win by the end of the game. It will also learn the best action for each state. In this work a 4-layer CNN network followed by 2 fully connected feedforward networks.

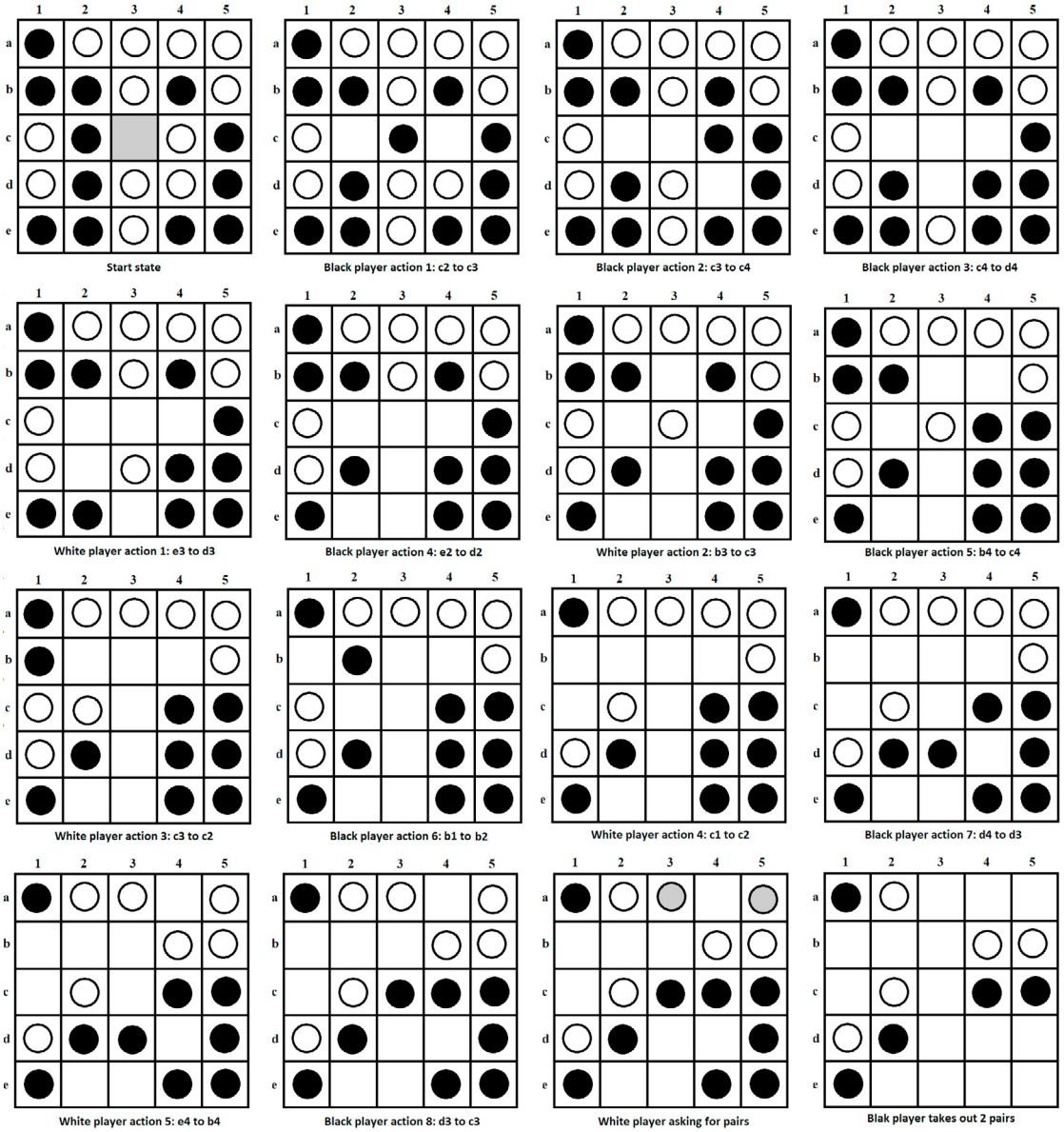

Fig. 2. Full moving stage (stage two) for 5x5 Seejeh game. Black player starts the stage.

4.3. Policy Iteration

This section describes the training process. Policy iteration algorithm 1 represents the learning process through self-play. Policy iteration algorithm starts with a random policy $\overrightarrow{\pi_t}$ and value since the neural network is initialized with random weights. A number of self-play games is played in each iteration. A fixed number of MCTS simulations starting from the current state s_t is performed in each turn of the game. A move is randomly picked from the improved policy. Now a training example consisting of state and policy is ready to run and find the final reward z_t at the end of the game. z_t is +1 for winning the game, and -1 otherwise. This completes the training example tuple $(s_t, \overrightarrow{\pi_t}, z_t)$. The obtained training examples are used to train the neural network at the end of each iteration. The current network and the old one are pitting against each other. The current network is updated to the new one if it wins more that a set of

threshold. Otherwise, another iteration is conducted to obtain a new training example. The network keeps improving as iterations goes by to get a better player.

4.4. Minimax Moving Player

The Minimax algorithm is used to play the second stage of the Seejeh game. The search space of this stage is much less than the search space of the positioning stage. The maximum branching factor of the first move of the first player is 4. The maximum move for any player that have a sequence of capture is also 4 since the player has to move the same tile that he starts moving. The number of tile $K$for black player p_1 at any time t is:

$$K_t = ((N * N - 1)/2) - M_t(P_1) \tag{2}$$

where, N is the board width. $M_t(P_1)$ is the number of captured tiles for player p_1 at time t. The number of empty spaces E_t in the board at anytime is:

$$E_t = (M_t(P_1) + M_t(P_2) + 1) \tag{3}$$

The maximum number of moving actions is $4 * E_t$. The depth of second stage depends on the positioning stage as well as the moving stage. According to Seejeh experts, the average number of actions in the second stage of 7X7 board is 50 to finish the game.

Algorithm 1 Policy Iteration Algorithm

```
1:  procedure POLICY ITERATION
2:      oldNN ← initNN()
3:      trainExamples ← []
4:      for i in [1, ..., numIteration] do
5:          for j in [1, ..., numGames] do
6:              s ← initialState()
7:              while true do
8:                  for k in [1, ..., numSim] do
9:                      MCTS(s, oldNN)
10:                     examples.add((s; π_s, _))
11:                     a* ~ π_s
12:                     s ← nextState(s; a*)
13:                     if stage1Ended(s) then
14:                         reward ← Minimax(s)
15:                         addRewards(examples, reward)
16:                     trainExamples.append(examples)
17:                 newNN ← trainNN(trainExamples)
18:                 if newNN beats oldNN > thresh then
19:                     oldNN ← newNN
20:     return oldNN
```

5. Conclusions

Seejeh board game is different from all other strategic board games. It has two stages; positioning and moving. Player place two tiles at each action in stage one. Player can have a sequence of moves in the second stage unlike Othello and Go. To the best of our knowledge we are the first who implements an agent that learns to play Seejeh. The agent learns to play Seejeh through pure self-play, without using any human knowledge. We used deep reinforcement learning agent for the first stage and a Minimax agent for the second stage.

References

[1] H. C. Bolton, "Seega, an egyptian game," Journal of American Folklore, pp. 132–134, 1890.

[2] R. S. Sutton and A. G. Barto, "Reinforcement learning: An introduction." MIT press, 2018.

[3] E. P. Manning, "Using resource-limited nash memory to improve an othello evaluation function," IEEE Transactions on Computational Intelligence and AI in Games, vol. 2, no. 1, pp. 40–53, 2010.

[4] W. Jaskowski and M. Szubert, "Coevolutionary cma-es for knowledge-free learning of game position evaluation," IEEE Transactions on Computational Intelligence and AI in Games, vol. 8, no. 4, pp. 389–401, 2016.

[5] S. Gelly and D. Silver, "Achieving master level play in 9 x 9 computer go." in AAAI, vol. 8, 2008, pp. 1537–1540.

[6] E. A. Heinz, "New self-play results in computer chess," in International Conference on Computers and Games. Springer, 2000, pp. 262–276.

[7] M. A. Wiering, "Self-play and using an expert to learn to play backgammon with temporal difference learning." JILSA, vol. 2, no. 2, pp. 57–68, 2010.

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

[9] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot et al., "Mastering the game of go with deep neural networks and tree search," nature, vol. 529, no. 7587, p. 484, 2016.

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

[11] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel et al., "Mastering chess and shogi by self-play with a general reinforcement learning algorithm," arXiv preprint arXiv:1712.01815, 2017.

[12] P. Liskowski, W. Jaskowski, and K. Krawiec, "Learning to play othello with deep neural networks," IEEE Transactions on Games, vol. 10, no. 4, pp. 354–364, 2018.

[13] T. Gonsalves, "Board games ai", in Advanced Methodologies and Technologies in Artificial Intelligence, Computer Simulation, and Human-Computer Interaction. IGI Global, 2019, pp. 68–80.

[14] M. Buro, "Experiments with multi-probcut and a new high-quality evaluation function for othello," Games in AI Research, pp. 77–96, 1997.