A Multi-agent Design of a Computer Player for Nine Men's Morris Board Game using Deep Reinforcement Learning

Jafar Abukhait
Department of Communications, Electronics and Computer Engineering
Tafila Technical University
Tafila, Jordan
jafar@ttu.edu.jo

Ahmad Aljaafreh
Department of Communications, Electronics and Computer Engineering
Tafila Technical University
Tafila, Jordan
a.aljaafreh@ttu.edu.jo

Naeem Al-Oudat
Department of Communications, Electronics and Computer Engineering
Tafila Technical University
Tafila, Jordan
naeemodat@ttu.edu.jo

***Abstract*—Deep Reinforcement Learning (DRL) has been recently deployed in many artificial intelligence applications, and game players are not an exception. Nine Men's Morris is a board game that has been addressed and implemented using different AI techniques. In this paper, a multi-agent design of a computer player is introduced that represents the placing, moving, and capturing phases of the Nine Men's Morris. This design is a self-play one that knows nothing about the game other than the rules. Monte Carlo Tree Search (MCTS) is combined with Convolutional Neural Network (CNN) in each agent to provide the DNN with the training data. This combination allows the DNN to play against itself and tune its weights to predict actions. This computer player design ensures a proper training of NN without any human dataset and can compete with expert humans in the board games .**

Keywords*—deep reinforcement learning, nine men's morris, MCTS, self-play, convolutional neural network.

I. INTRODUCTION

Artificial Intelligence techniques have been used widely in decision-making applications especially computer game playing. Computer games, especially board games, offer a great domain to test any new AI technique since they are formal, complex, highly constraint, and need decision making [1]. In general, AI techniques used in game playing can be categorized as follows [1]:

- Behavior authoring: which employs static ad-hoc representations.
- Tree search: which builds trees for sequence actions after searching the space of future actions.
- Supervised learning: which builds a learning model based on a labeled dataset.
- Reinforcement learning: in which agents learn good behaviors by cumulative rewards after interacting with its environment.
- Unsupervised learning: in which patterns are found based on non-labeled datasets.

Deep learning has also been deployed as a supervised learning technique by adding more hidden layers allowing recognizing more features [2]. In fact, supervised learning techniques happen to need high dimensional training dataset to come up with a decision making model [3]. This dataset is not always available for computer games and also tedious to build. On the other side, reinforcement learning has difficulties working on high dimensional state space to build a self learning decision making model. These limitations led to suggest deep reinforcement learning (DRL) which combines deep learning with reinforcement learning [4].

This technique trains the model on the state space generated from the reinforcement technique and tune the training model parameters based on the cumulative rewards [5]. DRL represents a great choice for computer game players since it implements self-play learning that compete with expert humans.

Nine Men's Morris is a board game that is played between two players. It requires a game board, as shown in Fig.1, and nine pieces for each player [6]. The pieces of each player should be different in shape or color than the opponent's pieces. Nine Men's Morris has different phases as described in Section III. It includes deep thinking and plans and thus; deploying AI techniques would be mandatory to implement a computer player for it at expert level.

In this paper, a multi-agent design of Nine Men's Morris game is proposed based on deep reinforcement learning. The proposed design implements a self-learning agent for both placing and moving phases of the game using both Convolutional Neural Network (CNN) and Monte Carlo Tree Search (MCTS). Another agent has been suggested for capturing pieces from the game board based on normal search. This design does not require a training dataset since it utilizes the reinforcement learning to provide the CNN with the game states and rewards in each phase of the game.

The rest of the paper is organized as follows: Section II summarizes the current computer player achievements and implementations of the Nine Men's Morris game and other board games. The game variations, phases, and strategies are described in Section III. Section IV gives a general description of our proposed method. In Section V, we provide the conclusions.

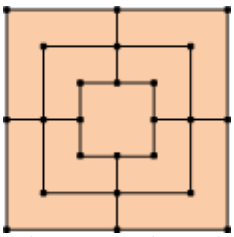

Fig.1. Nine Men's Morris game board.

II. RELATED WORKS AND IMPLEMENTATIONS

Board games are considered a great domain for testing AI techniques as they are formal, complex, highly constrained, and decision making environments [1]. Typically, intelligent agents are developed to play a game by implementing its rules and characteristics [7]. These agents are based on: search techniques, reinforcement learning, deep learning, or deep reinforcement learning.

In [8], a design and implementation for the board game Abalone have been proposed based on Alpha-Beta search. Minimax algorithm has been utilized for board games in [9]. It is also used in implementing Othello-playing programs [10].

Reinforcement learning (RL) has been used widely in developing self learning agents for computer games. In [11, 12] self-play learning agents have been utilized to create strategies. Go is a self-play learning agent that has been studied in [13]. A self-play techniques have been also used in [14] with Chess. Backgammon as a stochastic game was also proposed as a computer player using a self-play and knowledge based methods in [15]. Studies on mastering Chess, Shogi, and Go games, without human knowledge and based on reinforcement learning, have been introduced in [16, 17, 18].

Deep neural network were considered in developing intelligent agents for board games. In [19], deep neural network was used to master the Go game. Othello game was also studied in [20] using an experimental approach of several CNN-based deep neural networks. In [21], deep reinforcement learning has been introduced in developing board games. Both Chess and Doubles Pong games were developed using deep reinforcement learning [22, 23]. A multi-agent methodology has been introduced for generic board games in [24].

III. NINE MEN'S MORRIS STRATEGIES

Nine Men's Morris is an ancient board game dating back to 1400 B.C. [6]. It is a two-player, sequential, perfect information, deterministic, finite, and zero-sum game [25]. Nine Men's Morris is a solved game that is also known as Mills, Merrils, or Cowboy Checkers [26]. The game has also another three variations which are three, six, and twelve men's morris. The game board, shown in Fig. 1, is a grid that consists of three concentric squares and four segments connecting the midpoints of the three squares' sides together horizontally and vertically. This makes 24 intersections available for the two players to place their pieces.

Each player starts with nine pieces that are different from the other player pieces. The game starts by placing one piece at a time for each player and proceeds by moving a piece to a neighbor intersection at a time trying to make a mill, aligning three pieces along a horizontal or vertical line, which allows a player to capture or remove an opponent's piece from the board. The game ends either by reducing the pieces of an opponent to two pieces or leaving him without a legal move.

Nine Men's Morris has two basic phases [27]:

- Placing pieces: which starts on an empty board and each player places his pieces one piece at a time trying to make a mill. The player can capture any opponent's piece that is not in a mill if he succeed to make a mill. It is important to place pieces in versatile intersections and not concentrating the pieces in one side of the board [28]. Fig.2 shows two examples of placing phase.

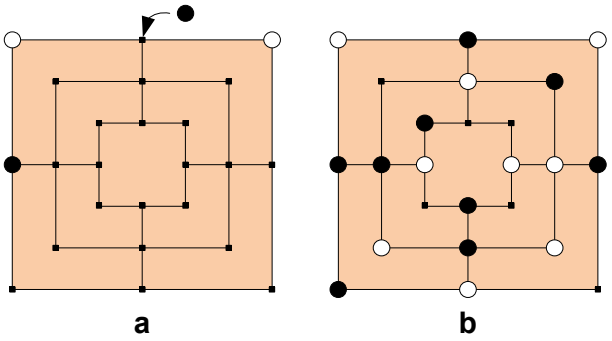

Fig.2. Placing phase examples. a) player with black pieces has to prevent his opponent from forming a mill by placing his piece between the two white pieces. b) a full placing phase has ended leaving six intersections empty.

- Moving pieces: in which each player moves on piece at a time to a neighbor and adjacent intersection trying to make a mill. The player can break his mill by moving one piece out of the mill and move it back after his opponent takes his turn. The player can move one piece back and forth between two mills and capturing opponent's piece at each turn. Fig. 3 shows and example of possible moving phase.

Capturing pieces is not a random process, player has to decide which is the best opponent's piece to capture in order to enable himself forming another mill. Fig.3 shows that the player with white pieces captured a black piece that enables him to form another mill at the middle-right vertical line at his turn..

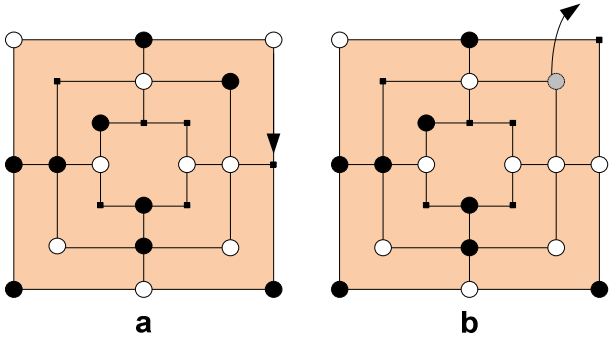

Fig.3. Moving phase example. a) player with white pieces moves one piece as shown by the arrow leading him to form a mill. b) player with white pieces decides to capture the black piece as shown by the arrow.

IV. NINE MEN'S MORRIS DESIGN

In this paper, we propose a multi-agent design for a computer player of the Nine Men's Morris game. The design architecture implements the two playing phases of the game. Three self-play learning agents of the game has been suggested: placing agent, moving agent, and capturing agent. The three agents communicate together in each playing phase and they are described as follows:

- Placing agent: which builds a self-play learning model to place the pieces on the game board. It generates an action by training a CNN and communicating with the other two agents.

- Moving agent: which builds a self-play learning model to move one piece to another valid location on the game board. It generates an action by training a CNN and communicating with the capturing agent.
- Capturing agent: which utilizes a normal search technique, since it has a limited search space, to capture (pound) one piece of the opponent's pieces from the game board. It does not generate an action and it can be called by either the moving or by placing agents in the playing phases.

The game board has been divided to 7x7 grid, as shown in Fig. 4, to ease the representation of the board state on the design. The state of the game board has been suggested accordingly to 7x7 array where each array cell represents a board intersection with a value of {-1, 1, 0, ∞} such that -1 for player 1 pieces, 1 for player 2 pieces, 0 for empty intersection, ∞ for unavailable intersection.

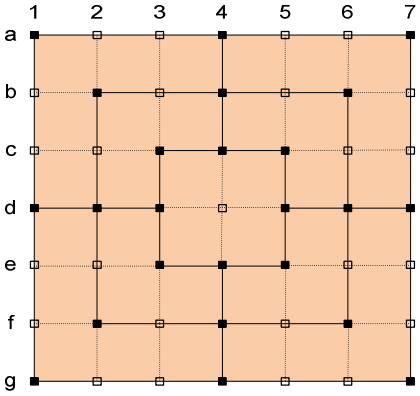

Fig. 4. 7x7 game board division for implementing game state.

A. Playing Agents

Each playing agent (placing, moving, or capturing) deploys DNN and MCTS in the training process. The neural network knows nothing about the Nine Men's Morris other than the rules. It plays against itself utilizing MCTS to predict actions and thus; update its parameters, in each iteration, to improve the performance.

The neural network f_θ has θ parameters which are the weights of the neural network. The state which represents the configuration of the board is the input of the DNN. The output is the probability vector over all possible actions ($p_\theta(s)$). It also gives another output which is the state value, which is a continuous number between 1and -1 ($v_\theta(s)$).

The DNN is initialized randomly at the beginning of the training phase. At each episode of the self-lay the DNN will be provided by training examples. These examples are a tuple of (s_t, π_t, z_t) where π_t is an updated estimate of the policy after MCTS starting from s_t and z_t is the final reward of the game from current player perspective. The final reward is obtained from the moving agent. The parameters θ of the neural network are updated through the training phase to minimize the error as in the equation below:

$$l = \sum_t (v_\theta(s_t) - z_t)^2 - \overrightarrow{\pi_t} . \log\left(\overrightarrow{p_\theta}(s_t)\right) \qquad (1)$$

By time, neural network will learn the best action and the value for each state. 3-layer CNN networks followed by 2 fully-connected feedforward networks have been deployed in each playing agent as shown in Fig.5.

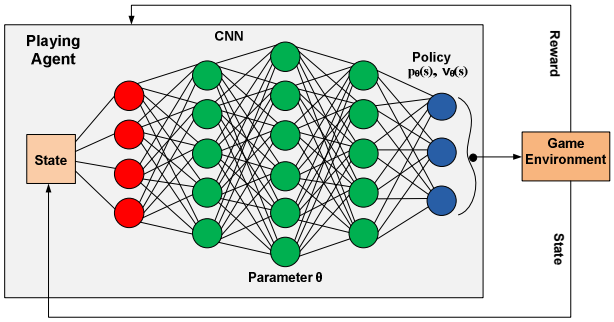

Fig.5. Playing agent architecture using CNN.

B. Placing Phase Policy

Placing pieces on the Nine Men's Morris game board does not happen systematically or randomly as mentioned in Section III. Good placing of pieces improves the chances of winning the game in the moving phase and thus; the placing agent calls the moving agent and could call the capturing agents in the placing phase as shown in Fig. 6. The training of the placing agent using CNN involves the training of the CNN in both the moving and capturing agents. The self-play learning process is implemented using the **policy iteration algorithm 1**.The algorithm starts with a random state value and policy π_t. At each iteration, a number of self-play games are played. At each turn of the game, several MCTS simulations are accomplished starting from the current state s_t. An action, which is placing a piece, is randomly selected from the improved policy. After that, a training example consisting of state and policy is ready to run to discover the final reward z_t at the end of the game. The moving agent is called to find the final reward after completing the placing phase. z_t is +1 for winning the game, and -1 otherwise. Then, the training example tuple (s_t, π_t,z_t) of each agent is complete. At the end of each iteration, these training examples are utilized to train the NN. After that, the new network competes against the old one. If the new one wins for a set of games then the network is updated to new one. Otherwise, iteration is performed to get a new training example. As iteration goes by the network improves.

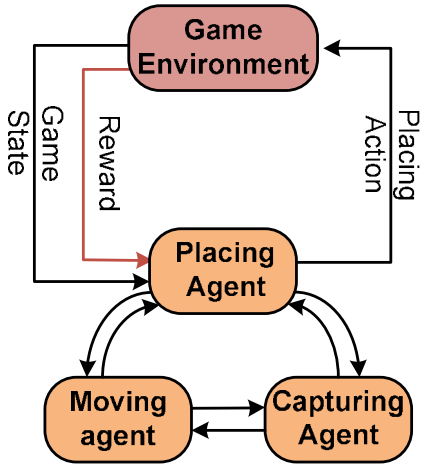

Fig. 6. The multi-agent architecture of the placing phase.

Algorithm 1 Policy Iteration Algorithm for Placement Phase

```
1: procedure POLICY ITERATION
2:     oldNN ← initNN()
3:     trainExamples ← [ ]
4:     for i in [1, ..., iterations] do
5:         for j in [1, ..., games] do
6:             s ← initialState()
7:             while true do
8:                 for k in [1, ..., simulations] do
9:                     MCTS(s, oldNN)
10:                examples.add((s, π_s, _))
11:                a* ~ π_s
12:                s ← nextState(s, a*)
13:                if phase1Ended(s) then
14:                    reward ← evalute(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
```

C. Moving Phase Policy

Moving phase on the Nine Men's Morris game board comes after the end of the placing phase. Capturing a piece happens as a result of possible move (forming a mill) and thus; moving agent could call the capturing agent as shown in Fig. 7. The self-play learning process for this phase is implemented as in the **policy iteration algorithm 1**. The algorithm steps areas in policy iteration algorithm 1since both of them implement a DRL self-play agent. The first difference is the input state. The moving phase input is a fully placed board performed by the placing agent. The second difference is the output. It is the moving-actions-probability vector and the value of the state. The action space for this player is the moving actions where each piece can move to one of its empty neighbor space. The third difference is the evaluation function for the end of the game as in line 14 in algorithm 1. The placing player call the moving playing for the evaluation of the end of the game. However, the moving player evaluates the end of the game by counting the difference in number of pieces for each color.

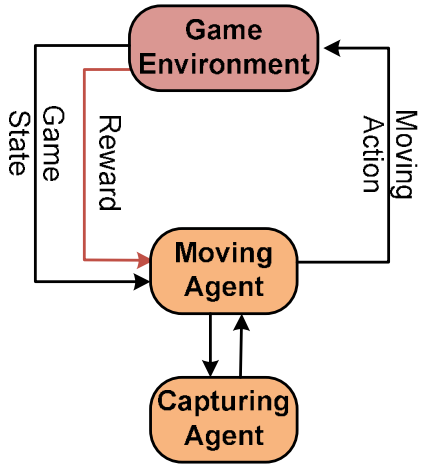

Fig. 7. The multi-agent architecture of the moving phase.

V. CONCLUSION

Nine Men's Morris is a board game that has two basic phases: placing and moving. In this paper, we proposed a multi-agent design for the game by suggesting two agents for the basic phases of the game. Another agent has also been suggested which implements capturing the pieces since it has strategy and does not happen systematically or randomly. The proposed design has been intended to play the game through self-play learning and does not need a training dataset or human knowledge. This has been implemented by combining and utilizing both the Convolutional Neural Network (CNN) and the Monte Carlo Tree Search (MCTS) in each agent. The three agents communicate with each other in the playing phases of the game.

REFERENCES

[1] G. N. Yannakakis, J. Togelius, Artificial Intelligence and Games, Springer, 2018, [online] Available: http://gameaibook.org.

[2] Y. LeCun, Y. Bengio, G. Hinton, "Deep learning", *Nature*, vol. 521, pp. 436-444, May 2015.

[3] S. Kotsiantis, S. Zaharakis, P. Pintelas, "Supervised Machine Learning: A Review of Classification Techniques", Informatica, vol. 31, pp. 249-268, 2007.

[4] V. Francois-Lavet, P. Henderson, R. Islam, M. G. Bellemare, and J. Pineau, "An introduction to deep reinforcement learning," Found. Trends Mach. Learning, vol. 11, no. 3-4, 2018.

[5] [12] R. S. Sutton and A. G. Barto, "Reinforcement learning: An introduction." MIT press, 2018.

[6] R. C. Bell, Board and table games from many civilizations. Courier Corporation, 1979, vol. 1.

[7] M. Rezende and L. Chaimowicz, "A Methodology for Creating Generic Game Playing Agents for Board Games," *2017 16th Brazilian Symposium on Computer Games and Digital Entertainment (SBGames)*, Curitiba, 2017, pp. 19-28.

[8] A. Papadopoulos, K. Toumpas, A. Chrysopoulos and P. A. Mitkas, "Exploring optimization strategies in board game Abalone for Alpha-Beta search," *2012 IEEE Conference on Computational Intelligence and Games (CIG)*, Granada, 2012, pp. 63-70.

[9] 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.

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

[11] 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.

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

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

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

[15] 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.

[16] 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.

[17] 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.

[18] 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.

[19] 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.

[20] 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.

[21] Xenou K., Chalkiadakis G., Afantenos S., "Deep Reinforcement Learning in Strategic Board Game Environments". In: Slavkovik M. (eds) Multi-Agent Systems. EUMAS 2018. Lecture Notes in Computer Science, vol 11450. Springer, Cham, 2019.

[22] M. Lai, "Giraffe: Using deep reinforcement learning to play chess," CoRR, vol. abs/1509.01549, 2015. [Online]. Available: http://arxiv.org/abs/1509.01549

[23] E. A. O. Diallo, A. Sugiyama and T. Sugawara, "Learning to Coordinate with Deep Reinforcement Learning in Doubles Pong Game," *2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA)*, Cancun, 2017, pp. 14-19.

[24] M. Rezende and L. Chaimowicz, "A Methodology for Creating Generic Game Playing Agents for Board Games," *2017 16th Brazilian Symposium on Computer Games and Digital Entertainment (SBGames)*, Curitiba, 2017, pp. 19-28.

[25] G. E. G´evay and G. Danner, "Calculating ultrastrong and extended solutions for Nine Men's Morris, Morabaraba, and Lasker Morris," IEEE Transactions on Computational Intelligence and AI in Games, vol. 8, no. 3, pp. 256–267, Sept 2016.

[26] F. Chesani, A. Galassi, M. Lippi and P. Mello, "Can Deep Networks Learn to Play by the Rules? A Case Study on Nine Men's Morris," in *IEEE Transactions on Games*, vol. 10, no. 4, pp. 344-353, Dec. 2018.

[27] R. Gasser, "Solving nine men's morris," Computational Intelligence, vol. 12, no. 1, pp. 24–41, 1996.

[28] Vedar, Erwin A.; Wei Tu; Elmer Lee. "Nine Men's Morris". GamesCrafters. University of California, Berkeley. Retrieved 2006-12-31.