Reinforcement Learning for Creating Evaluation Function using Convolutional Neural Network in Hex

Kei Takada
Graduate School of Information Science and Technology,
Hokkaido University, Sapporo, Japan
Email: takada@complex.ist.hokudai.ac.jp

Hiroyuki Iizuka
Graduate School of Information Science and Technology,
Hokkaido University, Sapporo, Japan
Email: iizuka@complex.ist.hokudai.ac.jp

Masahito Yamamoto
Graduate School of Information Science and Technology,
Hokkaido University, Sapporo, Japan
Email: masahito@complex.ist.hokudai.ac.jp

***Abstract*—An evaluation function in the board game decides the next move for computer AIs, and a high accurate evaluation function leads to a strong computer AI. Recently, the evaluation function using convolutional neural network(CNN) by supervised learning shows high evaluation accuracy. Supervised learning cannot exceed the teachers, but there must be a possibility to create a more accurate evaluation function by using reinforcement learning. In this paper, we proposed an evaluation function using CNN and reinforcement learning with games of self-play in Hex. The proposed evaluation function is tested with the previous evaluation function and world-champion program MoHex2.0. The results show that evaluation accuracy of the proposed evaluation function is higher than the previous evaluation functions, and proposed computer Hex algorithm EZO-CNN obtained a win rate of 60.0% against MoHex2.0 even though the search time of EZO-CNN is shorter than MoHex2.0.**

I. Introduction

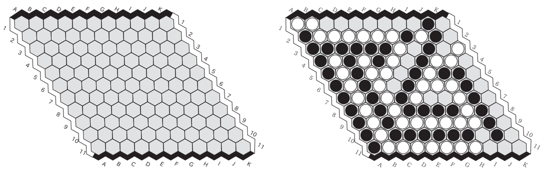
Learning algorithms and search algorithms have been proposed through board games, since the methods can be compared from easy-to-understand results, such as win or lose. The competitions between human professional and computer AI are attracting attention from all over the world, and the search algorithms for the best move in the current position are expected to be applicable to the field of artificial intelligence.

The evaluation function is the function that quantifies the advantage of the position for players. Since the computer algorithm plays the move evaluated as the best by the evaluation function, it is necessary to develop a highly accurate evaluation function to create a strong computer algorithm. The evaluation function calculates the evaluation values based on the features of the positions, which are usually obtained by a lot of efforts of human programmers. When the number of features increases, it becomes difficult to properly combine them manually. Therefore, a method of automatically adjusting weights using the large amount of game records [1] and a method of combining features using a neural network have been proposed [2]. Although these methods have shown that the created features are very useful, it is difficult to apply them to the games whose feature extraction is difficult.

Hex is a two-player board game invented by John Nash and is one of the board games included in the Computer Olympiad. In Hex, the board state is expressed by two board networks, and the network characteristics calculated from the board networks are used to evaluate positions [3]. In recent years, a move evaluation method of the candidate moves using CNN was proposed, and it showed that the move evaluation method using CNN is superior than the previous move evaluation method based on the network characteristics [4], [5]. This result shows the possibility that CNN can extract the features that cannot be represented by network characteristics, and it is also expected that the evaluation function of position using CNN is better than the previous evaluation function using network characteristics.

In order to create the high accurate evaluation function using CNN, reinforcement learning may be necessary because supervised learning cannot exceed the teachers. Supervised learning is used for creating the evaluation function using CNN in Go and Chess [6], [7]. AlphaGo combines the evaluation function using CNN (called *value network*) with a large number of simulations in order to improve the accuracy of the evaluating positions. If the evaluation accuracy is high enough, it is possible to reduce the simulations and computational cost for evaluating positions. By reducing the computational cost of the position evaluation, deeper game tree search becomes possible and better moves can be found.

In this paper, we propose a novel evaluation function using CNN in Hex, and the proposed evaluation function is trained by reinforcement learning with games of self-play. Supervised learning is commonly used to create an evaluation function using CNN, but supervised learning cannot exceed the teachers. We trained the proposed evaluation function with reinforcement learning using games of self-play in order to create the highly accurate evaluation function. We developed the computer Hex algorithm EZO-CNN based on minimax tree search using the proposed evaluation function. EZO-CNN was compared with the computer Hex algorithms using the previous evaluation function and world-champion algorithms in order to clarify that the proposed evaluation function is highly accurate. We created the evaluation function on 7×7, 9×9 and 11×11 and show the effectiveness of the proposed



(a) Hex gameboard (b) Game board showing a winning configuration for white

Fig. 1. An 11×11 Hex board. The top and bottom sides are assigned to the black player, and the left and right sides are assigned to the white player.

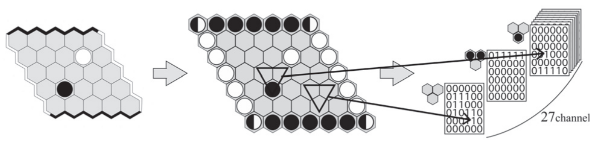


Fig. 2. Example of creating an input of the position. The left diagram shows positions on a 5×5 board. The middle diagram shows the sides in the left diagram with the stones. The cells on the corners are considered cells in which both black stones and white stones are placed. The right diagram is the input of the left diagram.The places corresponding to three mutually adjacency cell patterns become one.

evaluation function.

II. Hex

Hex is classified as a two-player, zero-sum and perfect information game. Hex was invented independently by Piet Hein and John Nash [8] and has been studied by many researches such as Shannon [9]. Currently, Hex is one of the board game of Computer Olympiads, and many methods have proposed through Hex.

A. Rules and Features

Hex is played on a rhombic board consisting of hexagonal cells. The game was developed around an $n \times m$ board (n, m are natural numbers), but an $n \times n$ board is generally used (Fig. 1(a) shows an 11×11 board). In recent years, 11×11 and 13×13 boards have been used in the Computer Olympiad [10]. The two players have uniformly colored pieces (e.g., black and white), and the game proceeds with players placing their stones by turn on empty cells. The two black opposing sides of the board are assigned to the black player, and the other two opposing sides are assigned to the white player. The goal of the game is to connect the two opposing sides by using the player's colored pieces: the black player wins if the black player successfully connects the black sides using black pieces, whereas the white player wins if the white player successfully connects the white sides using white pieces (Fig. 1(b)). In this paper, we used players with black as the first player and white as the second player.

It is shown that the first player has a winning strategy for an $n \times n$ board [11], the game cannot end in a draw [12], and the game is a PSPACE-complete problem [13]. A specific winning procedure for all first moves is proved on 9×9 or smaller boards [14].

B. Evaluation Methods

The position of the Hex board can be expressed as a network by regarding the cells as nodes and connecting adjacent nodes with a link (or edge) [3]. It is possible to quantify the features of positions by calculating network characteristics from the board networks. The evaluation functino can be designed on the basis of the network characteristics. The computer Hex algorithm which was the world-second champion in 2017 uses the evaluation function using 12 network characteristics proposed in the complex network [15]. Also, there is an evaluation function using an electric resistance model by regarding the board network as an electric circuit [3]. This electric resistance model is used by some computer Hex algorithms such as Wolve, Six, Hexy [16].

Move evaluation methods using CNN have been proposed without using the network characteristics [4], [5]. Move evaluation method is the function that evaluates the next all candidate moves. Previous study showed that candidate moves can be evaluated with higher evaluation accuracy by the move evaluation method using CNN than previous optimized move evaluation methods using the network characteristics.

C. Representation of Board Position for CNN

In this paper, we use the input proposed in the previous study [5]. In this section, we describe how to create the input to CNN proposed in the previous study.

In Hex strategy, it is important to consider the cell adjacency because the goal of Hex is to connect two opposite sides with own stones. In order to make it easier to learn the cell adjacency, three mutually adjacency cells are focused to create the input. The cell takes three states corresponding to the placement of the player's own stone, placement of the opponent's stone, the case where no stone is placed. Therefore, combinations of the states of the 3 mutually adjacent cells yield 27 patterns in total. The position is represented by 27 channels, and each pattern forms a channel. Fig. 2 shows an example of the how to create inputs from the position in which the black player makes a move. In each channel, the number corresponding to a specific channel pattern becomes one, and the others remain zero.

When the white player has the next move, we rotate the board by $180°$ from the axis of upper left cell to the lower right cell and swap the black and white cells because Hex board is symmetric. When the board is rotated, the left and right sides become the top and bottom sides, respectively. This configuration is obtained to eliminate the learning cost required for considering that the side to be connected by the player is different.

III. Proposed Method

In this section, we first describe the structure of the proposed evaluation function using CNN. Next, we describe a learning algorithm using games of self-play.

TABLE I
NETWORK STRUCTURE OF PROPOSED EVALUATION FUNCTION FOR 11×11 BOARD.

Layer type	Image size	Channel	Kernel size
Input	12×12	27	–
Conv 1	10×10	128	3×3
Conv 2	8×8	128	3×3
Conv 3	6×6	128	3×3
Conv 4	4×4	128	3×3
Conv 5	2×2	128	3×3
Conv 6	1×1	128	2×2
Full Connection	–	168	–
output	–	1	–

A. Proposed Evaluation Function using CNN

Our proposed evaluation function using CNN takes the board position as input and outputs the evaluation value of the position. Table I shows the network structure for our proposed model in 11×11 board. We use only convolutional layer in our CNN model. All activation functions in each convolutional layer are rectified linear units (ReLU) [17]. In output layer, the output values are transformed from 0 to 1 by using the sigmoid function. The proposed evaluation function aims to output 1 if the next player has the winning strategy and 0 if the next player has the losing position.

In this study, we create the evaluation function for 11×11, 9×9 and 7×7, and network structure is different according to each board size. Six convolutional layers are used in 11×11, but five convolutional layers and four convolutional layers are used in 9×9 and 7×7, respectively. By reducing the number of convolutional layers, the output size is set to one even for 7×7 and 9×9.

B. Learning Algorithm though Games of Self-Play

The proposed evaluation function is trained by reinforcement learning using games of self-play. Algorithm 1 shows our learning algorithm. We define the position at the t-th turn as s_t, the move in s_t as a_t, and evaluation value at s_t as $v(s_t)$. The player decides the move a_t at the position s_t based on minimax tree search of depth 1 using proposed evaluation function. The position-move pairs (s_t, a_t) appeared in the games are used for training proposed evaluation function. Also, the rotated position-move pairs $(s_t^{rotated}, a_t^{rotated})$ are used because rotated position-move pairs are essentially the same as the original pairs. The operation of the rotation is the same as the method described in Section Ⅱ C. Since learning various board states leads to improvement in generalization of proposed evaluation function, the techniques below are introduced to games of self-play.

Algorithm 1 Learning Algorithm

```
1:  function LEARNING
2:      for $epoch = 0$ to $E$ do
3:          Initialize training data set $\mathbf{D}$
4:          for $game = 0$ to $G$ do
5:              $p1$ uses the latest evaluation function
6:              $p2$ uses the latest or past evaluation function
7:              Randomly select which player is first
8:              Get random number $T$
9:              $t \leftarrow T$
10:             $s_t \leftarrow$ GETSTARTINGPOSITION($p1$,$p2$,$T$)
11:             while $not\ terminal\ s_t$ do
12:                 $p \leftarrow$ Next player selected from $p1$ and $p2$
13:                 $p$ searches best move $a_t$ at $s_t$
14:                 $(s_t, a_t)$ is added to $\mathbf{D}$
15:                 $(s_t^{rotated}, a_t^{rotated})$ is added to $\mathbf{D}$
16:                 $s_{t+1} \leftarrow$ Position playing $a_t$ at $s_t$
17:                 $t \leftarrow t + 1$
18:             end while
19:             ($s_t$,$\emptyset$) and ($s_t^{rotated}$,$\emptyset$) are added to $\mathbf{D}$
20:         end for
21:         Update the proposed evaluation function using $\mathbf{D}$
22:     end for
23: end function
24: function GETSTARTINGPOSITION($p1$,$p2$,$T$)
25:     $s_0 \leftarrow$ Initial position
26:     $s_1 \leftarrow$ Position playing the random move at $s_0$
27:     if $T$ is $1$ then
28:         return $s_T$
29:     end if
30:     for $i = 1$ to $T - 1$ do
31:         $p \leftarrow$ Next player selected from $p1$ or $p2$
32:         $p$ searches best move $a_i$ at $s_i$
33:         $s_{i+1} \leftarrow$ Position playing $a_i$ at $s_i$
34:     end for
35:     $s_T \leftarrow$ Position playing random move at $s_{T-1}$
36:     return $s_T$
37: end function
```

1) Addition of small random number to evaluation value: A small random number is added to the evaluation value outputted by the proposed evaluation function. The range of evaluation value is 0 to 1 because evaluation function uses sigmoid function, and the range of the random number to be added is -0.05 to 0.05. This range of random number is determined empirically, but by adding this random number, the player almost randomly plays the move if the player uses the proposed evaluation function which has not been learned.

2) Random move during games of self-play: A random move is played during games of self-play in order to train various positions. Game of self-play is done for T moves until the position s_T. T is the random integer number ($T > 0$). At the position s_T, the next move a_T is played randomly from all candidate moves. The game of self-play is done from the position s_{T+1} after playing a_T at s_T until the game terminates.

The positions appeared in game after the position s_{T+1} are used for training. The value range of T varies depending on the board size. The maximum values of T are 20, 40 and 60 in 7×7, 9×9 and 11×11, respectively. With each board size, even if the player plays move at random, there is a possibility that the game will end within T moves. If the game ends within T moves, the positions that appeared in the game are not used for learning.

3) Random move for the first move: The fist moves of games of self-play are selected randomly from the cells within two rows from the side of the board. The first move near the center of the board are advantageous for the first player in Hex. When starting the game from near the center, it is expected that many positions favorable to the first player appear. The evaluation function is required to accurately evaluate the position where it is difficult to determine which player is advantageous. Learning a lot of difficult positions can be expected to lead to accurate evaluation of the proposed evaluation function, so the games of self-play starts at cells within two rows from the side of the board. The number of cells to start the game are 40, 56 and 72 in 7×7, 9×9 and 11×11, respectively.

4) Using past evaluation function for games of self-play: In games of self-play, not only the latest proposed evaluation function but also the past evaluation function are used. The proposed evaluation function is stored every 10 epoch. During a single epoch, all the data contained in a dataset $\mathbf{D}$ is used once and the weights of the evaluation function is updated. $\mathbf{D}$ is a set that consists of position-move pairs appeared in the certain number of times G game of self-play. One player always uses the latest evaluation function. On the other hand, the evaluation function for another player is selected from the latest, the second or third latest evaluation functions. The selection of the evaluation function is performed randomly every game.

5) Determined move at the position where winning connection exists: Virtual Connection is a link between two cell groups that can be connected by playing the best move even if the player has the second move in the current positions [3]. Virtual connections are found by the algorithm called h-search, and virtual connection between two opposite sides is particular called winning connection [19]. If one player has winning connection found by h-search, the search by the evaluation function is no more performed. In the position where winning connection exists, the player with winning connection plays the winning move to connect two opposite sides, and the other player plays the most obstructive move.

C. Update Proposed Evaluation Function

The weights of the proposed evaluation function are updated by the stochastic gradient descent method by using the position-move pairs (s,a) appeared in games of self-play. The proposed evaluation function is trained to minimize the loss function $Loss$, which is defined as follows:

$$Loss = \sum_{(s,a)\in\mathbf{D}} loss(s,a), \tag{1}$$

$$loss(s,a) = \begin{cases} \log(1 - v(s)) & (s \text{ is terminal}) \\ (v(s) - v(s'(s,a)))^2 & \text{(otherwise)}, \end{cases} \tag{2}$$

(3)

where $\mathbf{D}$ is a set of position-move pairs (s,a) appeared in games of self-play; $v(s)$ is the output of proposed evaluation function at the position s; $v(s')$ is the output of the evaluation function at the position s' played the move a at the position s. It should be noted that the terminal positions included in $\mathbf{D}$ is the position that is defeated for the next player. So, the proposed evaluation function should output zero at the terminal positions included in $\mathbf{D}$.

IV. Experiments

The proposed evaluation functions were trained and created on 7×7, 9×9 and 11×11 by described learning algorithm in Section III. The obtained evaluation functions were evaluated from the two perspectives. One is whether the proposed evaluation functions are superior than our previous evaluation function and other world-champion programs. Another is evaluated in terms of search time.

First, the learning conditions of the proposed evaluation function are explained and the computer Hex used in the experiment are described. Next, we explain the experiment conditions for evaluating the proposed evaluation function. Finally, the winning percentages against MoHex2.0 are shown.

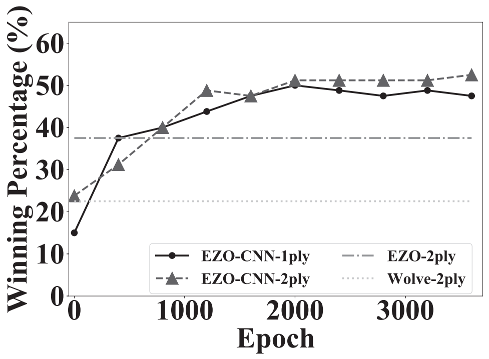
A. Learning Conditions

The number of game of self-play in a single epoch G were 500, 750 and 750 in 7×7, 9×9 and 11×11, respectively. We used a mini-batch size of 32 positions and Adam as the optimizer for training the proposed evaluation functions [23].

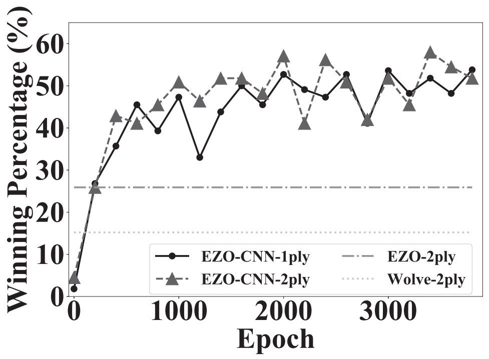
B. Computer Hex Algorithms

We used four computer Hex algorithms called EZO-CNN, EZO, Wolve and MoHex2.0. EZO-CNN is the proposed computer Hex algorithm based on minimax tree search with the proposed evaluation function. EZO is our previous algorithm that uses the 12 network characteristics for the evaluation function, which is optimized by *Minimax Tree Optimization*. It is the same evaluation function as the world-second-champion computer Hex algorithm in 2017 [15]. Wolve uses the evaluation function based on the electric resistance [3]. Wolve is the world-second-champion computer Hex algorithm in 2013 [20]. MoHex2.0 uses Monte Carlo tree search [21]. MoHex2.0 is the world-champion computer Hex algorithm and the strongest computer Hex algorithm among programs.

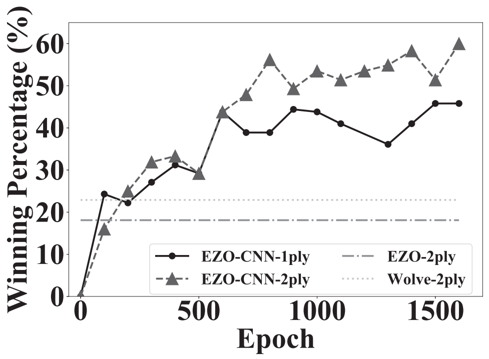
The computer Hex algorithms used in this paper were implemented on open-source Benzene framework [22]



(a) 7 × 7 board



(b) 9 × 9 board



(c) 11 × 11 board

Fig. 3. Winning percentages of each computer Hex algorithm against MoHex2.0.

C. Experiment Conditions

EZO-CNN and MoHex2.0 were compared from the direct games. The games between EZO-CNN and MoHex2.0 started at cells within two rows from the side of the board, and one game was played for the first and second players at each first move. The total first moves are 40, 56 and 72 in 7 × 7, 9 × 9 and 11 × 11, respectively, and there are 80, 112 and 144 games in 7 × 7, 9 × 9 and 11 × 11, respectively. The reason why the game is played near the edges is to reduce the advantage of the first player.

TABLE II
AVERAGE SEARCH TIME OF EACH COMPUTER ALGORITHM IN ONE GAME [SECOND / GAME].

Computer Hex Algorithm	Board Size		
	7 x 7	*9 x 9*	*11 x 11*
EZO-CNN-1ply	0.5	2.3	16.2
EZO-CNN-2ply	1.1	14.2	96.0
EZO-2ply	1.4	23.7	182.1
Wolve-2ply	0.9	13.1	101.0
MoHex2.0	29.5	145.0	312.3

In order to show which evaluation function is best, EZO-CNN, EZO and Wolve were indirectly compared through the winning percentages against MoHex2.0. EZO and Wolve played the games with MoHex2.0, respectively. The game conditions are the same as the game conditions between EZO-CNN and MoHex2.0.

Two types of EZO-CNN with different search depth were prepared to confirm whether the winning percentage improves by deeper search. The search depth of EZO-CNN-1ply and EZO-CNN-2ply are one and two, respectively. The search depth of EZO-2ply and Wolve-2ply are two. MoHex2.0 searches for up to 30 seconds per move.

D. Competition Results of Computer Hex Algorithms

Fig. 3 shows the winning percentage of EZO-CNN, EZO and Wolve against MoHex2.0 in the training process. The horizontal and vertical axes show the epoch number of learning and the winning percentage, respectively. The winning percentage of EZO and Wolve is constant because EZO and Wolve are not trained.

The winning percentages of EZO-CNN-1ply and EZO-CNN-2ply are higher than the EZO-2ply and Wolve-2ply, which shows that the evaluation accuracy of the proposed evaluation functions are higher than the previous evaluation function because the conditions are same except for the evaluation function. It is also confirmed that the maximum winning percentage of EZO-CNN-2ply against MoHex2.0 is 60% in 11 × 11, which means that EZO-CNN is stronger than the world-champion algorithm. In addition, especially in 11×11, it is found that the winning percentages of EZO-CNN-2ply are higher than the winning percentages of EZO-CNN-1ply. It is clarified that the deeper search using the proposed evaluation function improves the winning percentages.

Table II shows the approximate average search time of the games for each computer Hex algorithms. Despite the fact that the search time of EZO-CNN-2ply is shorter than MoHex2.0, EZO-CNN-2ply is stronger than MoHex2.0. It is clear that the proposed evaluation functions are very excellent.

The learning periods are about 2 days, 25 days and 27 days on 7 × 7, 9 × 9 and 11 × 11, respectively.

V. Discussion

In 7 × 7 and 9 × 9, the maximum winning percentages of EZO-CNN are about 50%, but this is considered to be a sufficiently high winning percentage. If the players are strong enough, the winning and losing is decided by only the first move because it is known that the winning strategy exists in Hex. The world-champion MoHex2.0 is strong enough, and MoHex2.0 may be able to play closer to the winning strategy on a small board size of 7×7 and 9×9. The winning percentage of 50% against MoHex2.0 means that EZO-CNN can also play the closer to the winning strategy. Furthermore, the maximum winning percentage of EZO-CNN often exceeds 50%. This result means that EZO-CNN can play the move to create the positions where it is difficult to play the winning strategy and can overturn the winning strategy of MoHex2.0.

The winning percentage of EZO-CNN-2ply is often higher than the winning percentage of EZO-CNN-1ply. It shows that the generalization ability of proposed evaluation function is very high. In general, the deeper search in the minimax tree search makes it possible to find better move because the closer to the terminal position, the easier the evaluating position by the evaluation function. However, there is a possibility of choosing a bad move with a deep search if generalization ability of the evaluation function is low. The search result is changed in the minimax tree search if the evaluation value of a certain position is changed, therefore the search result becomes a bad move if the evaluation function wrongly evaluates a certain position. Improving the winning percentage of EZO-CNN by deeper search shows that the evaluation accuracy and the generalization ability are high.

VI. Conclusion

In this paper, we proposed a novel evaluation function using CNN and reinforcement learning algorithm with games of self-play in Hex. We created the proposed evaluation function on 7×7, 9×9 and 11×11, and we developed the computer Hex algorithm EZO-CNN based on the minimax tree search using the proposed evaluation function. In order to clarify the evaluation accuracy of the proposed evaluation function is high, we tested EZO-CNN with the previous computer Hex algorithms. The result showed that the proposed evaluation function is superior to the previous evaluation functions. We also showed that EZO-CNN is stronger than the world-champion computer Hex algorithm called MoHex2.0 even though the search time of EZO-CNN is shorter than MoHex2.0. In the future, we will try to create the evaluation function with lager board size. We will also create the evaluation function by games of self-play with 2-ply search because it is known that the reinforcement learning with games of self-play by deep search improves the accuracy of the evaluation function.

References

[1] K. Hoki, and T. Kaneko, “Large-scale optimization for evaluation functions with minimax search”, Journal of Artificial Intelligence Research, vol. 49, pp. 527–568, 2014.

[2] G. Tesauro, “Temporal difference learning and TD-Gammon”, Communications of the ACM, vol. 38, pp.58–68, March 1995.

[3] V. V. Anshelevich, “A hierarchical approach to computer Hex”, Artificial Intelligence, vol. 134, no. 1–2, pp. 101–120, 2002.

[4] K. Young, G. Vasan, and R. B. Hayward, “NeuroHex: A Deep Q-Learning Hex Agent” Computer Games Workshop IJCAI, pp. 3–18, 2016.

[5] K. Takada, H. Iizuka, and M. Yamamoto, “Computer Hex using Move Evaluation Method based on Convolutional Neural Network” Computer Games Workshop IJCAI, in press, 2017.

[6] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, and M. Lanctot et al., “Mastering the game of go with deep neural networks and tree search,” Nature, vol. 529, no. 7587, pp. 484-489, 2016.

[7] O.E. David, N. Netanyahu, and L. Wolf, “DeepChess: End-to-End Deep Neural Network for Automatic Learning in Chess” Artificial Neural Networks and Machine Learning, ICANN 2016, pp.88–96, 2016.

[8] C. Browne, “Hex Strategy: Making the Right Connections” A. K. Peters, Natick, MA, 2000.

[9] C. E. Shannon, “Computers and automata,” Proceedings of the IRE, vol. 41, no. 10, pp. 1234–1241, 1953.

[10] R. B. Hayward, J. Pawlewicz, K. Takada, and T. van der Valk, “Mohex Wins 2015 Hex 11×11 and Hex 13×13 Tournaments” ICGA Journal, vol. 39, no. 1, pp. 60–64, 2017.

[11] J. Nash, “Some Games and Machines for Playing Them” D-1164, RAND, 1952.

[12] D. Gale,“The Game of Hex and the Brouwer Fixed-Point Theorem” The American Mathematical Monthly, vol. 86, no. 10, pp. 818–827, 1979.

[13] S. Even, and R. E. Tarjan, “A Combinatorial Problem Which Is Complete in Polynomial Space” Joutnal of ACM, vol. 23, no. 4, pp. 710–719, October 1976.

[14] J. Pawlewicz, and R. B. Hayward, “Scalable Parallel DFPN Search” Computers and Games CG2013 LNCS, vol. 8427, pp. 138–150, 2013.

[15] R. B. Hayward, and N. Weninger, “MOHEX WINS HEX 11X11 AND 13X13 TOURNAMENTS” unpublished.

[16] P. T. Henderson, “Playing and solving the game of Hex” PhD thesis, University of Alberta, 2010.

[17] X. Glorot, A. Bordes, and Y. Bengio, “Deep Sparse Rectifier Neural Networks” Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, vol. 15, pp. 315–323, April 2011.

[18] M. van der Ree, and M. Wiering, “Reinforcement learning in the game of Othello: Learning against a fixed opponent and learning from self-play” IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL), pp. 108–115, April 2013.

[19] J. Pawlewicz, R. Hayward, P. Henderson, and B. Arneson, “Stronger Virtual Connections in Hex” IEEE Transactions on Computational Intelligence and AI in Games, vol. 7, no. 2, pp. 156–166, 2015.

[20] B. Arneson, R. Hayward, and P. Henderson, “Wolve 2008 Wins Hex Tournament” ICGA Journal, vol. 32, no. 1, pp. 49–53, 2009.

[21] S.C. Huang, B. Arneson, R.B. Hayward, M. Muller, and J. Pawlewicz, “Mohex2.0: A pattern-based mcts hex player” Computer and Games, Springer LNCS 8427, pp. 60–71, 2014.

[22] B. Arneson, P. Henderson, and R. B. Hayward, “Benzene”, 2009–2012, http://benzene.sourceforge.net/.

[23] D. Kingma, and J. Ba, “Adam: A Method for Stochastic Optimization”, Proceedings of the 3rd International Conference for Learning Representations, 2015.