# Deep Convolutional Neural Network, Minorization-Maximization Algorithm, and Monte Carlo Tree Search on Block Go

Shi-Jim Yen[†1], Keng Wen Li[2], Chingnung Lin[3] and Jr-Chang Chen[4]

***Abstract*:** Block Go is similar to the game of Go. The game is introduced by Pro Zhang Xu **(張栩)**棋聖 at 2009. His purpose is to reduce the complexity of Go and let the game be suitable for children. The complexity of Block Go is around 10^45, which is between checker and Othello. In this paper, we will state the rule Go and analyses the complexity of Block. Then, the Block Go program is implemented with Monte Carlo Tree Search (MCTS), and minorization-maximization pattern database. We also apply Deep Convolutional Neural Network (DCNN) on Block Go. In the future, we will apply transfer learning to improve the DCNN of Block Go, based on the numerous Go game records.

**Keywords:** Block Go, Monte Carlo Tree Search, Minorization-maximization algorithm, Deep Convolutional Neural Network, Machine Learning.

## 1. Introduction

Block Go is a simplified version of Go. In 2009, Zhang Xu (張栩)棋聖 invented this game. Block Go has only nine blocks for each side and is played on a 13×13 Go board. This game is suitable for children and is popular in most Taiwanese children's Go colleges. In this paper, we use machine learning techniques and Monte Carlo Tree Search (MCTS) to develop Block Go programs. Section 2 describes the rule and complexity for Block Go. Section 3 introduces a native MCTS Block Go program. Section 4 and Section 5 apply machine learning techniques on this game. Finally, Section 6 makes the conclusion and states future works.

## 2. Rules and Complexity for Block Go

Block Go board is a 13×13 Go Board. There are nine blocks for each side of Block Go players, named Blue and Red. Each block is either one stone or a composed of stones. Seven blocks of them are the same as those in Tetris, named I, J, L, O, S, T and Z. The other two blocks are of the same kind, named B, composed by a single stone. Figure 1 and Figure 2 respectively show the initial board and a game, and the nine blocks. Figure 2 also shows the number of possible placement styles for one legal position. The rules are stated as follows.

1. In each ply, each player can place one block either on one of the four star positions that are (4,4), (4,10), (10,4) and (10,10), or on the positions adjacent to his/her own blocks. For example, in Figure 3, Blue (Red) can place a block on the white (green) points. Figure 4 illustrates illegal moves.
2. When all the 18 blocks are placed, the game is over. The counting of territory is the same as that of Go, and the player with more territory wins. The handicap is 3.5 points.
3. If one player's stones are within the opponent's territory and the stones have no eye(s), the stones are dead and belong to the opponent. Figure 1(right) is an example of a Block Go game. The blue stones in the left-down corner are dead.

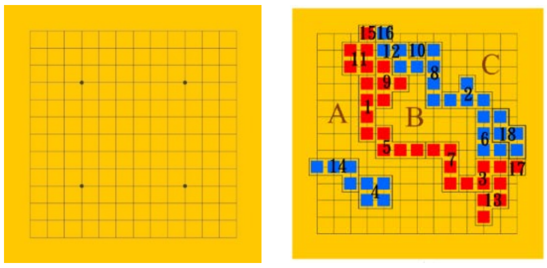
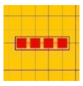
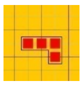
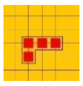
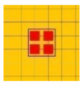


Figure 1. Initial board(left) and a game(right) for Block Go.

| Tetris name | I | J | L | O |
|---|---|---|---|---|
| possibility | 8 | 16 | 16 | 4 |
| Block |  |  |  |  |

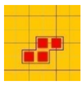
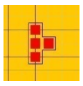
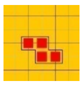
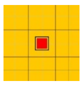
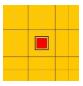
| S | T | Z | B | B |
|---|---|---|---|---|
| 16 | 16 | 16 | 1 | 1 |
|  |  |  |  |  |

Figure 2. The nine blocks and the numbers of their possible placement for one legal position.

[1] Shi-Jim Yen[†1], Keng-Wen Li[2], Chingnung[3] Lin are with the Department of Computer Science and Information Engineering, National Dong Hwa University, Hualien 97401, Taiwan (e-mail: sjyen@mail.ndhu.edu.tw, a499220381@gmail.com, jironglin@gmail.com).
[2] Jr-Chang Chen[4] is with the Department of Applied Mathematics, Chung Yuan Christian University, Taoyuan 32023, Taiwan. (e-mail: jcchen@cycu.edu.tw).

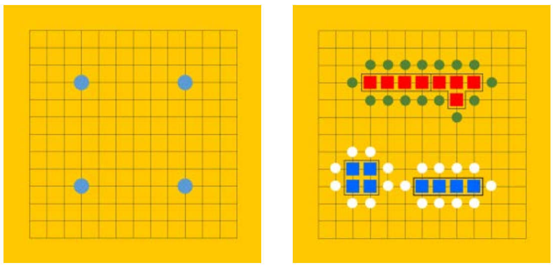


Figure 3. Legal positions..

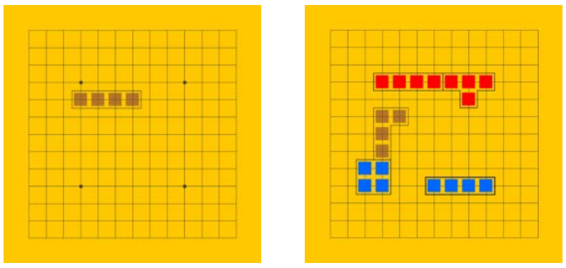


Figure 4. Illegal moves.

The complexity of Block Go is estimated as follows:

$$(77 \times 4) \times (77 \times 3) \times ((66 \times 10)^2) \times ((55 \times 17)^2) \times ((44 \times 22)^2) \times ((33 \times 26)^2) \times ((22 \times 30)^2) \times ((11 \times 33)^2) \times ((34)^2) \times ((35)^2) = 10^{45}.$$

For the first ply, the number of possible moves is 77, which is the possible block-placement method on one legal position for the nine blocks. Initially, there are 4 legal positions. Thus, the possibility of the first ply is $77 \times 4$. In the following plies, when a block is placed, the number of legal positions increases, but the number of the remaining blocks decreases. The complexity of Block Go is $10^{45}$ that is close to Nine Men's Morris ($10^{50}$) and Othello ($10^{58}$) [3].

## 3. A MCTS Block Go program

MCTS has four stages that are *selection*, *expansion*, *simulation* and *backpropagation* [1]. We use two machine learning techniques: minorization-maximization (MM) pattern database and deep learning on the expansion stage.

One of the differences between Block Go and Go is that only 18 plies in Block Go. Thus, the quality of the simulation is important. A bad simulation may not make any territory, because there are not enough moves to form the boundary. To solve this problem, we use MM patterns on the simulation stage. Furthermore, we also use three heuristic progressive-widening rules on the expansion stage:

1. For the first and second plies, they only play on the star positions, as shown in Figure 5(a).
2. For the third ply, the player can play only on the gray area in Figure 5(b).
3. For the forth ply, the player can play only on the gray area in Figure 5(c).

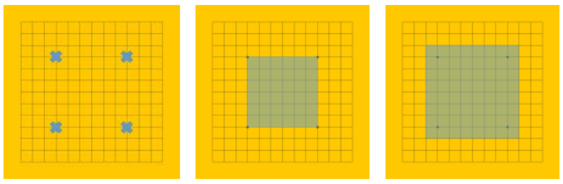


Figure 5. (a) Heuristic rule 1 (b) Heuristic rule 2 (c) Heuristic rule 3.

## 4. Machine Learning on Block Go

The MM pattern database is used for expansion and simulation. The MM algorithm is an iterative optimization method that exploits the convexity of a function in order to find their maximum or minimum. Remi Coulom applied MM algorithm on the Go move patterns [2]. Our MM pattern size is from 3 to 9. We collect 36,000 games from a Block Go APP. The strength of those games is between beginner and Go Pro. Table 1 shows the prediction rate with respect to the number of the training games. The prediction rate is not high. One of the reasons is the quality of the training games is not good enough.

Table 1. The prediction rate with respect to the number of training games

| Games | 13000 | 19000 | 25000 | 30000 | 36000 |
|---|---|---|---|---|---|
| Rate | 8.391 | 9.872 | 10.861 | 11.28 | 12.572 |

For studying the performance of MCTS Block Go programs. We use different settings for five Block Go programs as shown in Table 2. For example, DBH_MC is a MCTS program with MM pattern database and the three heuristic progressive-widening rules. Table 3 shows the competition results among the five programs.

Table 2. Five programs with different settings.

| Programs | pattern | Heuristic |
|---|---|---|
| DBH_MC | Yes | Yes |
| DB_MC | Yes | No |
| H_MC | No | Yes |
| MC | No | No |
| Rand | No | No |

Table 3. The result of the five programs with different seetings.

| | DBHr _MC | DB _MC | H_MC | MC | Rand |
|---|---|---|---|---|---|
| DBHr_MC | | 53.5% | 68.5% | 73.75% | 87.5% |
| DB_MC | | | 60.25% | 68.5% | 87.5% |
| Heur_MC | | | | 59.5% | 92.75% |
| MC | | | | | 91.25% |
| Rand | | | | | |

## 5. DCNN for Block Go

DCNN is very useful in computer Go. [8][9][10][11] This section describes a DCNN experiment on Block Go. Figure 6 is the DCNN for Block Go. The network architecture is similar to [8], but the input size is 13x13 with 23 planes instead of 19x19 with 26 planes. The K parallel softmax changes to 4 parallel softmax. A Block Go move is considered as 4 moves. For the 'B' move in Figure 2, we consider the other 3 moves as empty move. There are 23 planes as shown in Table 4.

Table 4. 23 planes of the DCNN

| | |
|---|---|
| 1 | Self-move |
| 2 | Opponent move |
| 3 | Empty |
| 4 | Boarder |
| 5-21 | 1st-17th move |
| 22 | All 1 plane |
| 23 | All 0 plane |

We collect 30000 Block Go games from internet. There are 18 positions in one game. Each position is processed with data augmentation with rotation at 90-degree intervals and horizontal/vertical flipping. We also delete some noise positions. Finally, there are 3680010 the train data positions and 129609 test data positions. We used Sigmoid Cross-Entropy as the loss function in DCNN. After 16 epoches, the best loss so far is 0.977121.

## 6. Conclusion

In this paper, we describe the rule of Block Go and introduce the complexity of Block Go. The rule is very simple, even simpler than the game of Go. The complexity is around $10^{45}$, which is between checker and Othello. Thus, Block Go may be a good testbed for computer games. We develop a MCTS Block Go program with MM pattern database. The experimental results show that the MM pattern is useful. We also implement DCNN on Block Go.

Because there are not many high quality pro Block Go game records, in the future, we will apply transfer learning to improve the DCNN of Block Go, based on the numerous 19×19 pro Go game records. Then it is possible to develop a strong Block Go program.

ACKNOWLEDGMENT

The authors would like to thank anonymous referees for their valuable comments in improving the overall quality of this paper, and Ministry of Science and Technology of Taiwan for financial support of this research under the contract numbers 104-2221-E-259 -009 -MY2.

## References

[1] Chaslot, G.M.J.-B., Winands, M.H.M., Uiterwijk, J.W.H.M., van den Herik, H.J., and Bouzy, B., "Progressive strategies for Monte-Carlo Tree Search," *New Mathematics and Natural Computation*, vol.4, no. 3, pp. 343–357, 2008.

[2] R. Coulom, "Computing Elo Ratings of Move Patterns in the Game of Go," *ICGA Journal*, vol. 30, 2007

[3] H. Jaap van den Herik, Jos W.H.M. Uiterwijk, Jack van Rinswijck, "Games solved: Now and in the future," *Artificial Intelligence*, vol. 134, pp. 277-311, 2002.

[4] Edward J. Powley, Peter I. Cowling, Daniel Whitehouse, "Information capture and reuse strategies in Monte Carlo Tree Search with applications to games of hidden information," *Artificial Intelligence*, vol. 217, pp. 92-116, 2014.

[5] Shi-Jim Yen, Cheng-Wei Chou, Jr-Chang Chen, I-Chen Wu, and Kuo-Yuan Kao, "Design and Implementation of Chinese Dark Chess Programs," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 7, issue 1, pp 66-74, 2015.

[6] Shi-Jim Yen, Tsan-Cheng Su and I-Chen Wu, "The TCGA 2011 Computer-Games Tournament," *ICGA Journal*, vol. 34, no. 2, 2011, pp. 108–110.

[7] Russell, S. and Norvig, P., *Artificial Intelligence: A Modern Approach* 2/e, Prentice Hall, 2003.

[8] Yuandong Tian and Yan Zhu, "Better Computer Go Player with Neural Network and Long-term Prediction", ICLR, 2016.

[9] David Silver, Aja Huang, Christopher J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel and Demis Hassabis, "Mastering the game of Go with deep neural networks and tree search", Nature, 2016, Pages 484-503, Vol. 529.

[10] Chang-Shing Lee, Mei-Hui Wang, Shi-Jim Yen, Ting-Han Wei, I-Chen Wu, Ping-Chiang Chou, Chun-Hsun Chou, Ming-Wan Wang, and Tai-Hsiung Yang, "Human vs. Computer Go: Review and Prospect", IEEE Computational Intelligence Magazine (IEEE CIM), Vol. 11, No. 3, pp. :67-72, August 2016.

[11] Chris J Maddison, Aja Huang, Ilya Sutskever, and David Silver, "Move evaluation in go using deep convolutional neural networks", In International Conference on Learning Representations, 2015.

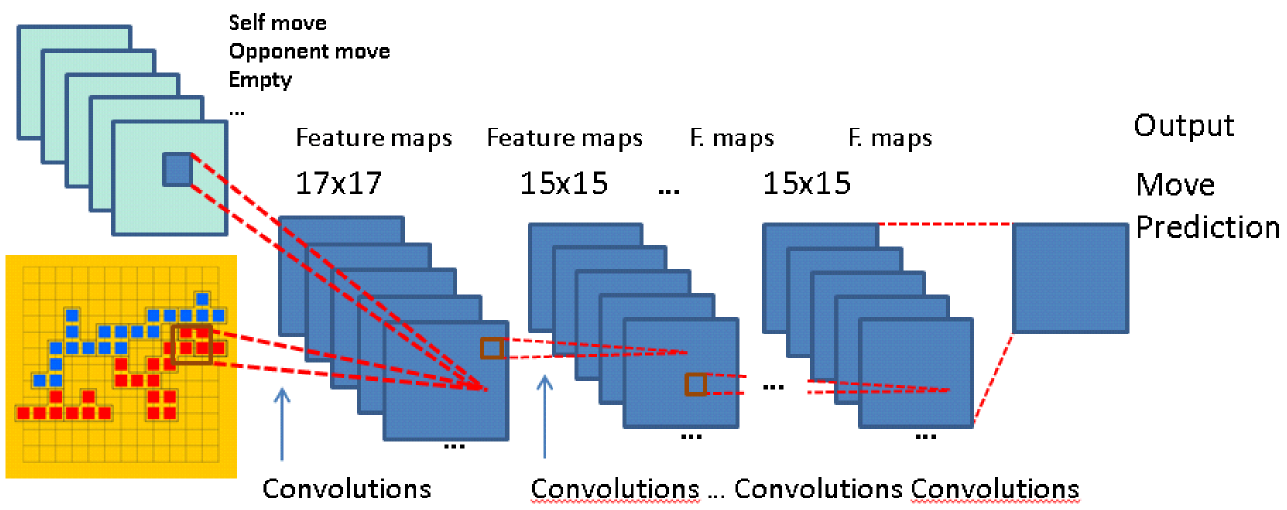


Figure 6. DCNN for Block Go