An Improved AlphaZero Self-Play Strategy for Dots and Boxes

Ming-Rung Li[1] **Chu-Chun Chi**[2] **Ke-Ching Chang**[3]

Abstract

Dots and Boxes is a two-player combinatorial game. In this paper, we apply the well-known AlphaZero algorithm to the Dots and Boxes game. Furthermore, we propose to use the game termination strategy during AlphaZero self-play to reduce the training time. To elaborate, we train 2 models for 2 different board sizes, which are 3 × 3 boxes and 5 × 5 boxes, respectively, and do the comparison with them. Finally, we confirm the effectiveness of our AlphaZero agent using a pure MCTS agent. The result shows that our AlphaZero agent achieves a winning rate of over 93% against the MCTS agent.

keywords: Dots and Boxes, Monte-Carlo Tree Search, AlphaZero, Computer Games

1. Motivation

Dots and Boxes is a classical occupation game. It requires two players to take over grids as many as possible. In contrast with the majority of board games in that players will take turns to place pieces, Dots and Boxes allows the player who just captures boxes to keep drawing another line. Due to these special game rules, Dots and Boxes has a more complicated game process than other games, thus having a hard time training an agent. As a result, it is hard for traditional Artificial Intelligence to develop a strong agent with a complex gaming system since the strongest program consists of profound knowledge in the specific domain, masterful techniques, and strict and efficient evaluation functions.

In 2017, DeepMind proposed a novel algorithm – AlphaZero. It can achieve superhuman performance in many computer games by only knowing the rules of the games. AlphaZero trains the agent by reinforcement learning based on the experience from self-play without any additional domain knowledge. Therefore, in this paper, we try to utilize the AlphaZero algorithm to train models with a Dots and Boxes gaming system so that we can merely provide them with game rules and have higher performance. Apart from this, we make use of the game termination strategy which assigns a new terminal state so that the game can stop earlier than the original Dots and Boxes game.

2. Background

2.1 Dots and Boxes

The board of Dots and Boxes is composed of many grids, while each box is formed by 4 adjacent dots. Each player takes turns drawing a horizontal or vertical line between 2 adjacent dots. To be more specific, it is prohibited if any player repeatedly draws a line at the same place. Moreover, a player can capture a specific box if he or she has the last edge to form the box, then must draw another line until there is no other box captured by the player. In the end, a Dots and Boxes game is over when all boxes are occupied. The winner is the one who owns more boxes eventually. In addition, it is not required to fill a box if one of the players has the

[1] National Yang Ming Chiao Tung University (Email: a0988282303@gmail.com)

[2] National Yang Ming Chiao Tung University (Email: chuchun1231@gmail.com)

[3] National Yang Ming Chiao Tung University (Email: cassie610512@gmail.com)

ability to do so. In figure 1, we demonstrate part of the gameplay process of the 3 × 3 Dots and Boxes game.

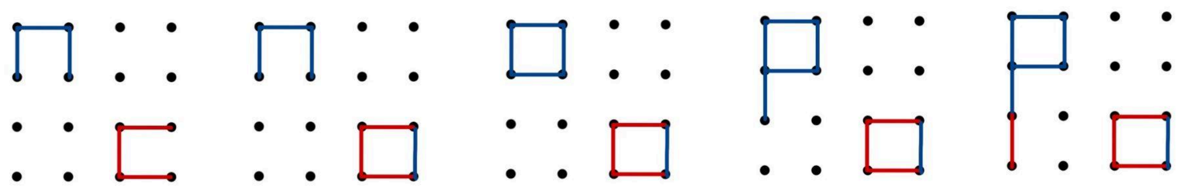

Figure 1 The gameplay process of the 3 × 3 Dots and Boxes

2.1.1 Impartial Game

Dots and Boxes is an impartial game in which moves are available only depending on the board configuration, instead of who the current player is. Therefore, there is no need to record which action is made by which player. The only thing that needed to be recorded is to keep the current board state.

2.1.2 Fairness

Being the first player or second player will have a different advantage. As a result, we will define one turn to be 2 games in which players will take turns to be the first player for fairness.

2.2 Related Works

Nowadays, there are many brilliant studies about Dots and Boxes. For example, Barker J et al [2] use Alpha-Beta search to solve 4 × 5 Dots and Boxes, the result is a tie given optimal play. Lex Weaver and Terry Bossomaier have attempted to utilize artificial neural network models to train AI agents to play Dots and Boxes games [8], but the results are not ideal. In this section, we will quickly review some previous works most related to our research.

2.2.1 Monte-Carlo Tree Search

Monte-Carlo Tree Search (MCTS) [3] plays an essential part in artificial intelligence. It aims to find the best decision based on the different game rules through random sampling among decision pools and choose the one with the highest estimated value. Each iteration of MCTS has the following 4 phases:

(1) Selection: Traverse recursively from root node to leaf node according to the UCT formula, which balances between exploitation and exploration.

(2) Expansion: After arriving at the leaf node, randomly choose one legal node as the new leaf node of the tree.

(3) Simulation: Simulate the condition of randomly placing pieces starting from the expanded node.

(4) Backpropagation: update the simulation result along the previously selected path from the leaf node to the root node

For Dots and Boxes games, the previous research [10] proposes an innovative board representation with an improved MCTS agent. Besides, some previous research applies early playout terminations [5, 7] on MCTS playout which terminate a playout earlier than the end of the game when it is likely to win, lose, or draw. In [7], they used an evaluation function to determine the winner of the playout instead of letting the random playout of MCTS run until the end of the game in Amazon, Havannah, and Breakthrough. In [5], for Chinese Dark Chess, they create detection rules based on a material combination and terminate the game once the likely outcome for a material combination is win, loss, or draw.

2.2.2 AlphaZero

AlphaZero [4] is a generic reinforcement learning algorithm, which can apply to various board games. In contrast with other traditional AI models that must be proficient with domain knowledge, AlphaZero has spectacular performance with merely knowing basic game rules. The training process of AlphaZero is composed of 2 parts, which are self-play and neural network training. It will first implement self-play, and then neural network training can take advantage of the trajectories from self-play to learn the value and policy. To elaborate, self-play is to perform MCTS with the current board state as the root, while MCTS follows the predicted value from the policy and value network.

There is also another study [9] about applying AlphaZero to the Dots and Boxes game for 5 × 5 grids, while they do not utilize the game termination strategy. As for the performance of the AlphaZero agent, it demonstrates superior strength against the pure MCTS agent with the win rate being 100%.

3. Method

In this section, we will reveal the detailed design of the AlphaZero framework and the optimization application on our Dots and Boxes gaming system. We will uncover how we apply the termination strategy in Dots and Boxes in section 3.1. And illustrate some parameters to build a well-performed AlphaZero agent in 3.2 and 3.3. There are previous works about early playout terminations in MCTS algorithms [5, 7], however, we innovatively apply this technique to the terminal state in AlphaZero self-play for the Dots and Boxes game, creating a straightforward evaluation function. By being the first attempt to apply this technique to AlphaZero self-play for Dots and Boxes games, we can accelerate the convergence of Dots and Boxes games compared with the rate of orthodox game rules.

3.1 Game Termination Strategy

According to the game rules, a game ends when all the boxes are taken by players. Therefore, it is intuitive to set the terminal condition of the AlphaZero self-play to the last step of the game. Take board size being 3 × 3 grids for example. When the step count reaches 24, the terminal condition occurs and the game is over. However, the game outcome may be determined earlier than reaching the last step. That is, if the board size is 3 × 3 boxes and the total available boxes are 9, once a player owns 5 boxes, the game is over and that player wins. Based on this game characteristic, we propose to adopt the game termination strategy, in which the self-play is terminated when one of the players captures more than half of the total number of boxes.

3.2 Self-Play

As mentioned above, self-play will implement the MCTS algorithm to generate trajectories for neural network training. In our design, when the number of the newly generated sampling data reaches a threshold, 16384 in our case, the training of both the policy and value network models will start to process. Specifically, the generated state will be stored in a replay buffer and be sampled later for training.

The self-playing data will be generated depending on the latest policy and value network models during the selection phases. Then, in expansion and simulation, the neural networks will serve as the evaluation function. In our case, after finishing 400 iterations, we pick up the action with maximum visited counts or merely randomly sample among the available actions.

The reason for using the latest neural network is because it benefits us to get a well-performed neural network. If we keep updating continuously, we can get more different models, thus converging faster.

Additionally, the diversity of self-playing data plays a crucial part. To increase the data diversity, we use two methods that were suggested by the AlphaZero algorithm. One is sampling the visited counts distribution mentioned above, the other is using Dirichlet noises during root policy. The formula is presented in (1).

$$P(s, a) = (1 - \epsilon)\lambda a + \epsilon\eta a \qquad \textbf{(1)}$$

From (1), P(s, a) is the probability of policy, while λ and η are the policy generated from the MCTS and Dirichlet noises, respectively. Moreover, ϵ is 0.25 in our framework.

3.3 Network Models

To get the well-trained network model for predicting the policy and value, we will use the self-playing trajectories to train the network models. The policy network model will output the MCTS visited counts of each available position, and the output of the value network will be the result of each run of the game, which are +1, 0, -1 meaning win, draw, and lose, respectively.

In terms of the design of the neural network, we employ ResNet as the backbone, while we utilize $3 \times 2 \times 12$ to represent the current board state when the board size is 3×3 boxes and $3 \times 2 \times 30$ for the board size being 5×5 boxes. Furthermore, take board size being 3×3, as an example. The input for the neural network will be $3 \times 2 \times 12$ as mentioned above. 2×12 suggests that there are 24 available actions, while 3 uncovers that we keep 3 channels of the current board state, which are the non-occupied lines for now, occupied actions for now, and who the current player is, respectively. Similarly, we can deduce that there will be $3 \times 2 \times 30$ board state for the board size being 5×5 since there are 3 channels $\times$ (2×15 available actions).

To let the output of the neural network be closer to the result of MCTS, we need to minimize the loss function. As for the loss function, it is shown in (2). It is composed of the cross-entropy of policy and the mean square error of value with a regularization term, which is used in case of over-fitting.

$$l = (z - v)^2 - \pi T \log p + c \mathbin{//} \theta \mathbin{//}^2 \qquad \textbf{(2)}$$

Besides, z is the true game result, while v indicates the predicted value from the policy network. π represents the truly MCTS visited counts distribution, and p suggests the predicted policy of the policy network. In addition, $c \mathbin{//} \theta \mathbin{//}^2$ is just the regularization term.

In our case, we originally let our network models have 4 channels suggesting the occupied actions from the first player, the occupied actions from the second player, the available actions, and the current player. However, in light of the attribute of an impartial game, we then make the channel size 3 representing the occupied actions from both players, the available actions, and the current player.

4. Experiments

In this section, we utilize the AlphaZero framework [1] from Computer Games and Intelligence Lab to conduct the experiments. The AlphaZero agent decides its actions based on the policy and value network mentioned in section 3.3.

4.1 The Training of 3 × 3 Boxes

We compare the proposed AlphaZero agent with a baseline MCTS agent with 1000 simulation counts. It is worth noting that to increase the variety of the opening of the games, we use a random function to determine the first step of the game. The result may be influenced by this particular factor.

As mentioned in section 3.2, we applied a game termination strategy to the game. At first, we set the terminal condition of the AlphaZero agent to the finish of the last step of the game, which is the 24th step in a 3 × 3 boxes game. Then, we find it takes a long time to converge. Thus we modified the terminal condition such that when one of the players captures more than half of the total number of boxes, we stop the game and give the rewards. After applying the new terminal condition, the average game steps in self-play drop from 24 steps to 20.64 steps, and the average of minimum game steps in each iteration are merely 15 steps. Figure 2 shows the decreasing loss in both termination strategies.

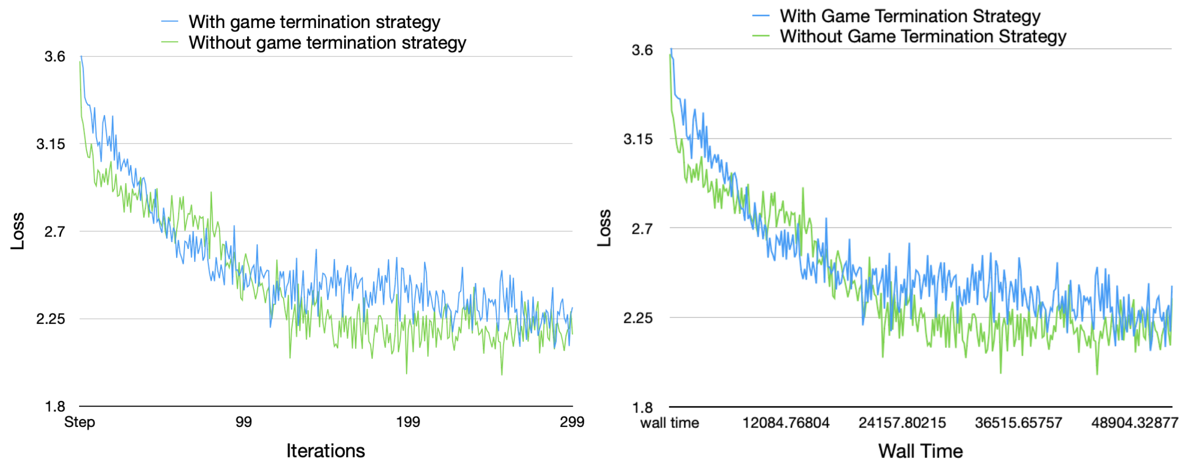

Figure 2 The loss of AlphaZero self-playing before and after we apply the game termination strategy in a) different iterations, b) different wall time.

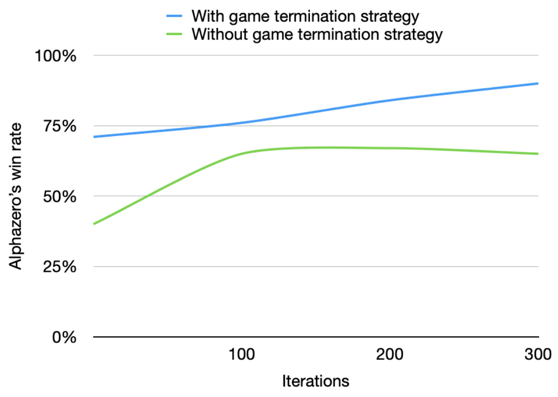

Figure 3 the convergence of the training process before and after we applied the new terminal rule. The x-axis of the chart displays the iteration of AlphaZero agent and the values along the y-axis indicate the win rate of the AlphaZero agent v.s. MCTS agent (with simulation counts equal to 1000) after playing for 100 games

Table 1 Pearson Correlation between training iterations and AlphaZero's win rate.

	Without game termination strategy	With game termination strategy
Pearson Correlation	.756	.998**
p value	.244	.002

From Figure 3 we can observe that after applying the game termination strategy, not only does the total number of steps decrease but the win rate of the AlphaZero agent is generally higher in different training iterations. Also, from the result in Table 1, the Pearson Correlation between training iterations and AlphaZero's win rate is statistically significant ($p < .05$) only when we apply the game termination strategy. Finally, comparing the outcomes of our AlphaZero agent versus the MCTS agent, it is evident that our AlphaZero agent achieved a higher win rate than the MCTS agent.

4.2 The Training of 5 × 5 Boxes

Given the experience of training a 3 × 3 Dots and Boxes game, we directly apply the game termination strategy that we mentioned in section 3.1 on the training of 5 × 5 boxes. The average game steps in self-play drop from 60 steps to 52.9 steps, and the average of minimum game steps in each iteration becomes 38.2 steps. This significantly reduces the game state complexity in self-play.

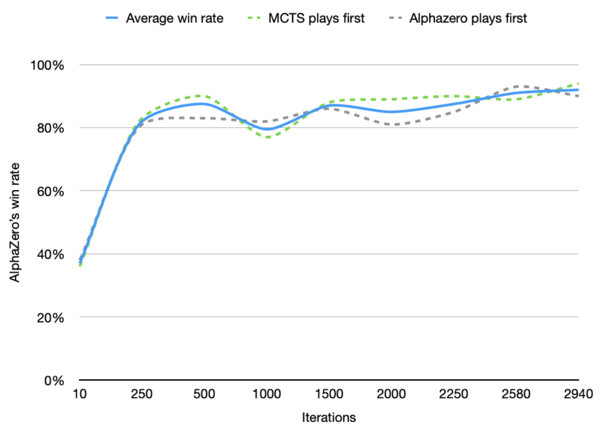

Figure 4 The win rate of different training iterations after we applied the game termination strategy. The x-axis of the chart displays the iteration of the AlphaZero agent and the y-axis indicate the win rate of the AlphaZero-vs-MCTS agent after playing for 200 games

Table 2 The Kendall tau rank correlation coefficient between the training iterations and AlphaZero's win rate.

Kendall tau rank correlation coefficient	.704**
p value	.009

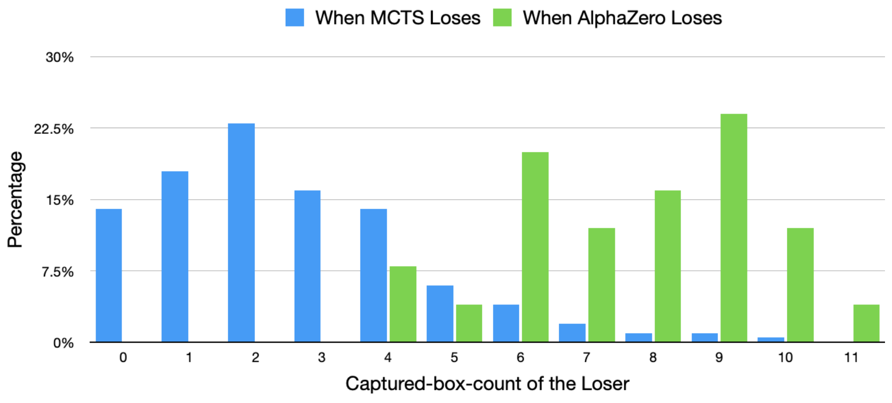

Figure 5 The percentage of the captured box count of the MCTS agent and the AlphaZero agent, respectively, while the opponent reaches the box count of 13 (the winning box count in a 5 × 5 Dots and Boxes game). In other words, the percentage of the number of boxes the agent has captured while it lost the game.

Based on the competition results in Figure 4 and 5, the AlphaZero agent demonstrated superior performance compared to the baseline MCTS agent (with simulation counts = 1000), as reflected by its higher win rate and the fact that it tends to capture more boxes while it loses the game compared to the MCTS agent. Also, given the Kendall tau rank correlation coefficient between the training iterations and AlphaZero's win rate, the result is statistically significant ($p < .05$). Eventually, The win rate of the AlphaZero agent reaches over 93% in the final training iteration.

5. Conclusion

In this paper, we perform research applying the AlphaZero algorithm on Dots and Boxes gaming systems with board sizes being 3 × 3 and 5 × 5 boxes and wish their performance to be greater than the MCTS agents. In addition, we utilize the game termination strategy in AlphaZero self-play for reducing the training time. By contrast with the traditional Dots and Boxes game rules, the new strategy allows the terminal state to be reached earlier, and thus has positive effects on the efficiency and convergence. In the end, the outcomes of the experiments meet our expectations. Obviously, our AlphaZero agents do have a higher win rate than the pure MCTS agents. Moreover, the AlphaZero agents have a much higher probability of owning more boxes when losing the game than the MCTS ones. In conclusion, the AlphaZero agents with the game termination strategy outperform the MCTS agents in each aspect we observe.

6. Future Works

Since there are previous studies [2] that solve 4 × 5 Dots and Boxes using Alpha-Beta search, we would also like to compare our AlphaZero agent with the theoretical results. However, since the theoretical value of Dots and Boxes is not publicly available, we look forward to calculating the theoretical value of Dots and Boxes ourselves in the future and comparing it with other results.

Furthermore, we wish to get a deeper insight into Dots and Boxes that we want to train an agent whose heuristic is based on the MuZero algorithm. Besides, we will further observe the difference between the AlphaZero agent and the MuZero[6] agent.

Acknowledgements

Thanks for all the support from Computer Games and Intelligence Lab, NYCU CS. This work was performed under the supervision of Hung Guei and Prof. I-Chen Wu. Besides, the AlphaZero framework used is partially supported by the National Science and Technology Council (NSTC) of the Republic of China (Taiwan) under Grant Number 110-2221-E-A49-067-MY3.

References

[1] Cheng, Yu-Hsuan, ”The Design of a MuZero Framework.”National Yang Ming Chiao Tung University, Hsinchu City, 2021.

[2] Barker, J., & Korf, R. (2021). Solving Dots-And-Boxes. *Proceedings of the AAAI Conference on Artificial Intelligence*, *26*(1), 414-419.

[3] Browne, Cameron B., et al. ”A survey of monte carlo tree search methods.” Computational Intelligence and AI in Games, IEEE Transactions on 4.1 (2012): 1-43.

[4] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy Lillicrap, Karen Simonyan, Demis Hassabis, Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm, arXiv:1712.01815, 2017.

[5] Hsueh, CH., Wu, IC., Tseng, WJ., Yen, SJ., Chen, JC. (2015). Strength Improvement and Analysis for an MCTS-Based Chinese Dark Chess Program. In: Plaat, A., van den Herik, J., Kosters, W. (eds) Advances in Computer Games. ACG 2015. Lecture Notes in Computer Science(), vol 9525. Springer, Cham.

[6] Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, Timothy Lillicrap, David Silver, Mastering Atari, Go, Chess and Shogi by Planning with a Learned Model, arXiv:1911.08265, 2020.

[7] Lorentz, R. (2015). Early Playout Termination in MCTS. In: Plaat, A., van den Herik, J., Kosters, W. (eds) Advances in Computer Games. ACG 2015. Lecture Notes in Computer Science(), vol 9525. Springer, Cham.

[8] L. Weaver and T. Bossomaier. Evolution of Neural Networks to Play the Game of Dots-and-Boxes. In Alife V: Poster Presentations, May 16-18 1996, pages 43-50.

[9] Yajun Zhang, Shuqin Li, Xiaojun Xiong, ”A Study on the Game System of Dots and Boxes Based on Reinforcement Learning”, IEEE 2019.

[10] Yimeng Zhuang, Shuqin Li, Tom Vincent Peters, Chenguang Zhang, ”Improving Monte-Carlo Tree Search for Dots-and-Boxes with a novel board representation and artificial neural networks .”, IEEE CIG 2015.