Artificial Intelligence – 2nd Assignment Two-Agents Board Game (Congklak) Reinforcement Learning

Cindy Aprilia (Matricole number : 286702)

***Abstract*—Congklak is an ingenious board game from Indonesia, that is played by two player. This game have many variations from across the regions. Reinforcement Learning was added to previously enhanced game in first assignment to further enhanced the AI performance.**

I. Introduction

THIS report was made to report how the game was made and worked. Along with this report, presented as well the screenshots and statistic of the algorithm to play these games.

II. Definition

A. LITERATURE REVIEW

1. CONGKLAK

   Congklak is Indonesian game which playing board is made from wood with either 5, 6, 7 or 9 holes in both sides (The varies number of hole is due to variations from island to island), plus 2 homebases (1 for each player). There are 2 players on this game, which compete to get as many shells as possible in their homebase. In order to win, the player either need to have the most shells in his/her homebase (*menang biji)* or to be the last person to run out of shells on your side of the board (*menang jalan*).
   For further instruction on how to play the game, please refer to :
   https://www.expat.or.id/info/congklakinstructions.html
   Watch the visualization in here :
   https://www.youtube.com/watch?v=pxT4BbsdybY

B. ALGORITHM [1]

1. MIN-MAX

   Min-Max Search is used in AI to implement systems that can play zero-sum perfect-information games. Typically, this algorithm is implemented for two players board game.
   It is an adaptation of hill climbing heuristic search alternating the choice of what state to choose next, according to the min or the max of the heuristic evaluation of the board position, depending which player turn is.
   The idea is that while player A is trying to maximize the evaluation of the position, the opponent is believed to be minimizing the same evaluation.

2. MIN-MAX WITH CHANGEABLE DEPTH

   Minimax is a depth-first, depth-limited search procedure, and is the prevaling strategy for searching game trees. Minimax searches down to a certain depth, and treats the nodes at that depth as if they were terminal nodes, invoking a heuristic function (called a static evaluation function) to determine their

3. ALPHA-BETA PRUNING

   Alpha–beta pruning is a search algorithm that seeks to decrease the number of nodes that are evaluated by the minimax algorithm in its search tree. The algorithm maintains two values, alpha and beta, which respectively represent the minimum score that the maximizing player is assured of and the maximum score that the minimizing player is assured of. For example, when searching the agent realize move "A" will improve the player's position, and then agent will continues to look for moves to make sure a better one hasn't been missed. It found that move "B" is also a good move, but the player then realizes that it will allow the opponent to force checkmate in two moves. Thus, other outcomes from playing move B no longer need to be considered since the opponent can force a win. So, it will take Move "A" as it's suggested moved.
   values.

4. MIN-MAX WITH DECREASE/INCREASE ONLY MOVE

   Because the heuristic of congklak game is to make the agent either Menang Jalan or Menang Mati, which means the best heuristic of this game is cutting the exchange of turn. Therefore, we can not stipulate that the exploring only increasing or decreasing move will help, as at many moment player will need to get the number that help them to secure Home Base

[1]For the further usage how these algorithm was implemented, please refer to jupyter notebook and python code remarks

instead of get the highest and lowest. Due to conflict with heuristic, the feature of increase and decrease move distance was skipped.

5. REINFORCEMENT LEARNING

Reinforcement learning is one of three basic machine learning paradigms, alongside supervised learning and unsupervised learning. Reinforcement learning (RL) is an area of machine learning concerned with how intelligent agents ought to take actions in an environment in order to maximize the notion of cumulative reward.

6. BASIC Q-LEARNING

Q-learning is a model-free reinforcement learning algorithm to learn the value of an action in a particular state. It does not require a model of the environment (hence "model-free"), and it can handle problems with stochastic transitions and rewards without requiring adaptations.

C. IMPLEMENTATION

1. USER EXPERIENCE AND CODE LIMITATION

1.1. ENHANCEMENT OF PREVIOUS MIN-MAX

User can choose number 1 to show the representation of congklak game, and compare the performance.(Question 1a, 1b, 1c)

The objective of this menu are

- to compare which depth of the min-max is the best for learning purpose
- whether cut variance added value for game performance, either for training, playing, or both

The features of this menu :

- User can choose how many times the code will be run for testing purpose

There are static function as toggle for :

- Cut Variance
- Explore depth
- Hole and Beads Combination
- Randomize a few first moves (Usually use 1 as default value to represent our user is the 2nd user, with 1st user is only random unheuristic player)
- Alpha-Beta is added as default to min-max
- The testing result not only printed on cmd, but automatically saved to csv for evaluation purpose as well.

Extra capabilities :

- The code is capable to accommodate cross playing between min-max, random, and partly random agents
- The agents can be multiples, however author lock it into 2 only to represent real conglak game that only can be played by 2 people
- Board size can be changed with toggle.
- Can be shown whether the player is menang jalan or menang mati
- Because the heuristic of this game is to make the agent either Menang Jalan or Menang Mati, which means the best heuristic of this game is cutting the exchange of turn. Therefore, we can not stipulate that the exploring only increasing or decreasing move will help, as at many moment player will need to get the number that help them to secure HomeBase instead of get the highest and lowest. Due to conflict with heuristic, the feature of increase and decrease move distance was skipped.

1.2. TRAINING DATASET

User can choose number 2 to show the representation of training (Question 2a)

The objective of this menu are to train the agent with Reinforcement Learning and Q to cut the searching time and resource in the future

The features of this menu :

- User can choose how many times the code will be run for training purpose
- The dataset will be saved in model folder
- To accomodate different size of board, different model file will be created according to the size of the board.

1.3. PLAYING WITH REINFORCEMENT LEARNING

User can choose number 3 to show the representation of playing with dataset (Question 2b)

The objective of this menu are to show the representation of agent that played with Reinforcement Learning and Q.

The features of this menu :

- When the move have been added to the dataset, agent would not do exploration anymore, and changed into searching from dataset.
- When the agents play the game, that not exists in dataset, it will played it with min-max (enforced with alpha-beta pruning, recommended depth-according to our testing and defining, and cut variance), and then saved to dataset, to broader the dataset

2. CODE EXPLANATION

2.1. Main (Main.ipynb)

The starting page of this project. In this page, the user can choose whether they want to simulate congklak

game without learning agent, training congklak agents, or playing congklak game with trained agent.. The system will run the next steps according to the choice. (see C.1)

2.2. CongklakBoard.py
This is the page where the congklak board was emulated. As the enhancement, this board's size can be changed to accommodate various regions game.
In here, there are function to check whether the move valid or not.

2.3. CongklakAgent.py
This is the page containing the playing agent for congklak game.

2.4. CongklakSearchingAgent.py
This is the page where min-max and its enhancement (AlphaBeta, Variance Pruning, Depth, etc) was placed.

2.5. CongklakTrainingAgent.py
This is the page where reinforcement learning was placed.

2.6. train_congklak.py
This is the page where Congklak Board, Agent, and Regressor was combined, in order to generate dataset.

2.7. play_congklak.py
This the page where Congklak Games was held. User can choose whether the agents is naïve or learned. Including the level of the depth.

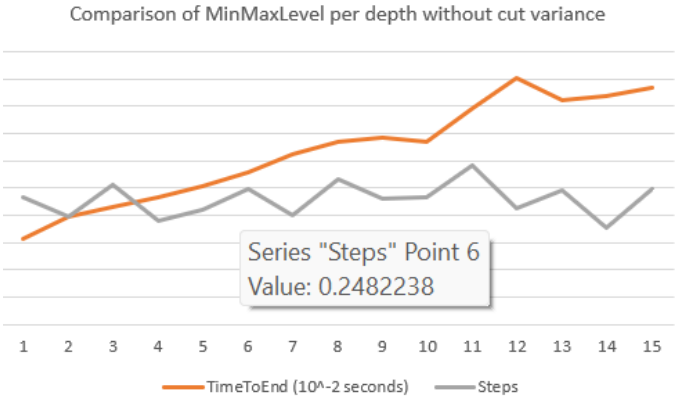
2.8. Requirements.txt
This is the text file where required library python was listed.
pip -r requirements.txt

D. *RESULTS*

1. DEPTH AND ALPHA-BETA
Stated in table below is the average time performance and steps taken from 33.381 games
(Complete result in ANNEX 1).
Which consists of combination between fully min-max and partly min-max (to simulate one of the player using random move, to make higher variation of the result).
33381 games was consists of 15 level depth testing

SUMMARY		
Depth	Steps	Performance time (in 10^{-6} seconds)
1	12.23446	156391.3
2	12.19716	196608.3
3	12.25533	215121.8
4	12.19005	232026.8
5	12.20915	252411.7
6	12.24822	279223.5
7	12.20071	310910.8
8	12.26736	335293.2
9	12.2307	342202.6
10	12.23252	333778.2
11	12.29155	397290.2
12	12.21163	452920.5
13	12.24659	411194.3
14	12.1762	418524
15	12.2475	435070.9

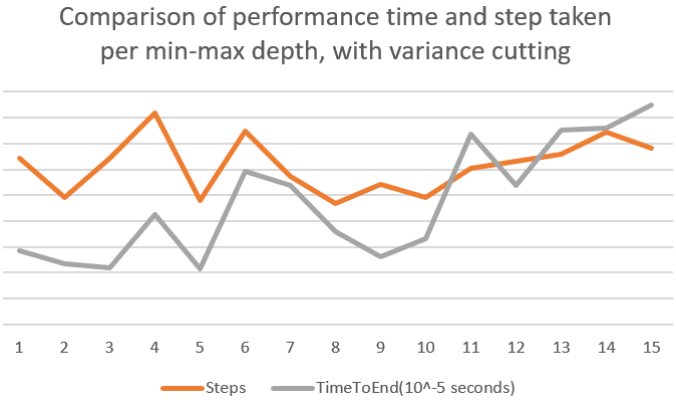
From the result we can seen that in each depth level raising, there are raise of performance time as well. However, the number of steps require to finish the game is not always in line with that. Therefore, without cut variance, author can decide that level 4 is the best, as the steps taken the third smaller of 15 depths (12.19 steps in average), and the performance time was 4th smallest (0.23 seconds).
However, for training agent, author would recommend level 14, in order to decrease future playing steps (12.17 steps in average).



2. VARIANCE VARIATION
Stated in table below is the average time performance and steps taken from 34.092 games.
(Complete result in ANNEX 2).
Which consists of combination between fully min-max and partly min-max (to simulate one of the player using random move, to make higher variation of the result).
34092 games was consists of 15 level depth testing

SUMMARY		
Depth	Steps	Performance Time (10^{-6} seconds)
1	32.14227	142527.3
2	24.628	117285
3	32.16864	109769.9
4	40.795	212386.7
5	23.99045	106943
6	37.35385	297093.4
7	28.61363	268836.6
8	23.41125	180442.9
9	27.18147	130289.5
10	24.5	166807.6
11	30.26888	367431.3
12	31.57293	268408.6
13	32.83288	376501.4
14	37.18982	379648.7
15	34.15631	424756.4

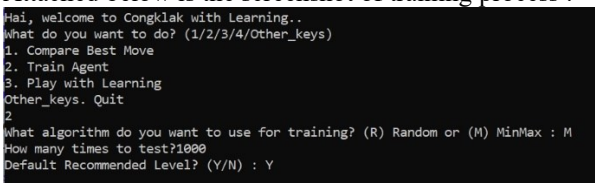
From the result we can seen that in each level depth raising, there are difference time performance and steps taken. However, the number of steps require to finish the game is not always in line with the depth level. Therefore, without variance, author can decide that level 5 is the best, as the steps taken the seconds smaller of 15 depths (23.99 step and the time is 0.106 seconds to finish.

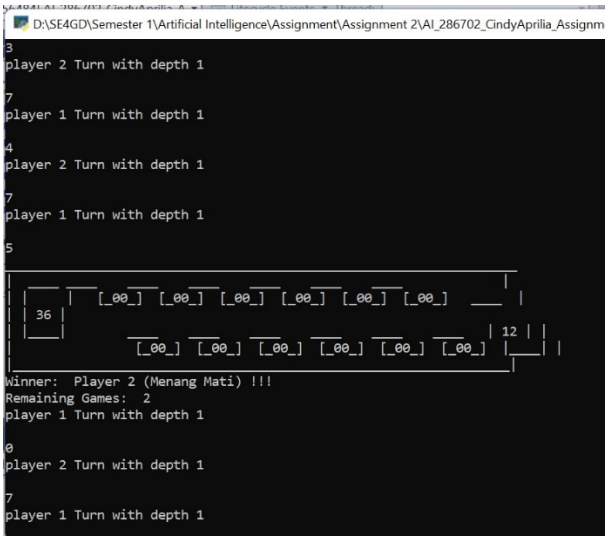


3. TRAINING

If user choose number 2, then the system will trained the dataset with min-max (depth can be stated). (Complete result in ANNEX 3).

Attached below is the screenshot of training process :





4. LEARNED AGENT

4.1. αβMinMax-L2 VS αβMinMax-LearnedL1

From 1000 games played, all win-ed by player 2 (player with reinforcement learned L1), with average time is 0.0314 seconds
(Complete result in ANNEX 4).

4.2. αβMinMax-L4 VS αβMinMax-LearnedL1

From 1000 games played, all win-ed by player 2 (player with reinforcement learned L1), with average time is 0.03057 seconds
(Complete result in ANNEX 5).

4.3. αβMinMax-L2 VS αβMinMax-LearnedL4

From 1000 games played, all win-ed by player 2 (player with reinforcement learned L4), with average time is 0.02961 seconds
(Complete result in ANNEX 6).

4.4. αβMinMax-L4 VS αβMinMax-LearnedL4

From 1000 games played, all win-ed by player 2 (player with reinforcement learned L4), with average time is 0.02669 seconds
(Complete result in ANNEX 7).

III. CONCLUSION

From the testing attempts, author can conclude that certain depth will raise the performance of Congklak Agent, however the same thing can not be said for variance cutting as well. As the heuristic of Congklak Rule, indeed do not depend on how near the each group and cost of the move. But, more on how to prevent the enemy to take turns. Therefore, checking the depth deeper to predict the enemy movement, will be much more useful and work significantly faster than adding this feature.

Regarding reinforcement learning, author found out that in board game simulation reinforcement learned agents always

win when compete with agent without learning. The time performance was faster as well.

When we use min-max with best depth to be ground of the training, the agents trained with that level works 0.01 faster than level reinforcement learning of min-max L1.

IV. Acknowledgment

The authors gratefully acknowledge works of harukigonai and mkgray, as their previous works greatly inspired this project.

V. References

[1] Caianello P. and Stillo G., “Introduction of AI”, 2022
[2] Caianello P. and Stillo G., “Heuristic Search”, 2022
[3] Caianello P. and Stillo G., “Agent of Ontology”, 2022
[4] Caianello P. and Stillo G., “Min-Max”, 2022
[5] Caianello P. and Stillo G., “Learning”, 2022

VI. ANNEX

There are several annex attached with this report inside excel file.
The annex was not included inside the pdf because due to the number of testing (more than one thousands per scenario), there are not enough spaces.

1. Depth Changing Testing of MinMaxAlphaBeta Without Variance Pruning
2. Depth Changing Testing of MinMaxAlphaBeta With Variance Pruning
3. Training Dataset
4. Testing of MinMaxAlphaBeta With Trained Agent : αβMinMax-L2 VS αβMinMax-LearnedL1
5. Testing of MinMaxAlphaBeta With Trained Agent : αβMinMax-L4 VS αβMinMax-LearnedL1
6. Testing of MinMaxAlphaBeta With Trained Agent : αβMinMax-L2 VS αβMinMax-LearnedL2
7. Testing of MinMaxAlphaBeta With Trained Agent : αβMinMax-L4 VS αβMinMax-LearnedL2