Artificial Intelligence of Knight Isolation Game with Multimax and Q-Learning Algorithm

Kaiyan Jin, Lu Han, Runxin Peng, Yan Gong
Department of Electrical Engineering and Computer Science, Syracuse University
{kjin07,lhan11,rpeng07,ygong20}@syr.edu

***Index Terms*—Knight Isolation, Multimax, Q-learning**

I. Introduction

Knight Isolation [1] is a game that Knights can move to any open cell that is 2-rows and 1-column or 2-columns and 1-row away from their current position on the board (like "L"). . This project will address to optimize the decision of AI in this game using two types of algorithm: Multimax(tree search algorithm) and Q-learning(machine learning model). The result of this experiment will tell how the two models imporve the decisions of the AI. Github.

II. Game

Knight Isolation is a type of adversarial game. It can be taken as simplified variation of international chess. In this game, every player controls one knight to move rotately. In each their own turn, it is only allowed to move to 8 directions as suggested in Fig3. 2 or 3 people is required to play this adversarial game. Whenever either player occupies a cell, that cell becomes blocked for the remainder of the game. The first one find no legal move to play is the loser of the game, the rest of players are winners of the game.

The main data structure is to record the current position and turn in this game and the history of move. We design it as the variable named "board state"(1d array) that is designed to save 3 types of above information: the history position of each player(1x (width x height)), the current turn (1x1) and the current positon of each chess(1x3).

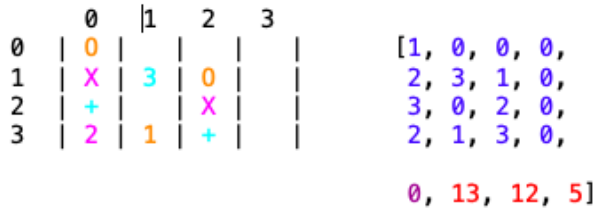

Fig. 1: Data Structure of Board State (Right Side) and Image printed from it (Left Side)

There are 2 types of variation in our experiment: board size and number of players. In the traditional implementation of this game: there are typical 2 players. In our projects, we design to play a more complicated way - 3 players. For the board size, we design it with 5 different size. The Large size means rapidly increase of computing. It will challenge the feasibility of the algorithm.

Our project is based on this design [2]. We revise the number of players, board ,model of this open source project. We also optimize some data structure and logic of the game.

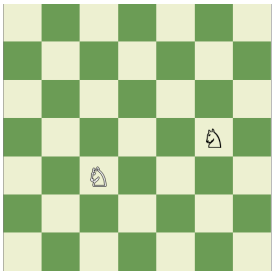

Fig. 2: Initial of the Game

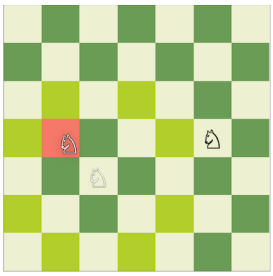

Fig. 3: Move of the Game

III. Tree search

The selected Tree search of this experiment is Multimax. Multinax is a variation of Minmax tree Search[3]. The game tree of Knight Isolation is suggested as Figure 4. The Multimax can be taken as a brute-force search. We search tree based on current board state and stop until some player has no choice to move. The loser of that case get utility of -2. The winner of that case get utility of +1. For each player, the optimal solution has been made. In other words, every player will choose the solution of maximum utility in the tree search. Finally after depth breath search of whole possibility, the AI player make a decision based on above algorihm in that step. The concept of choosing of Non-AI player based on utility is the virtual conclusion, we are supposed to make

clear the difference between that and the real decision made by randomness.

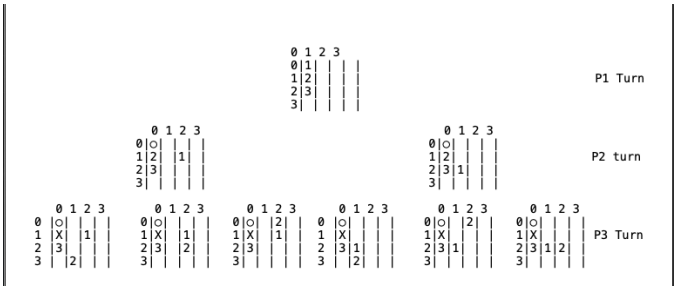

Fig. 4: The Game tree of Knight Isolation:P1 has two legal move, P2 has three legal move in each branch

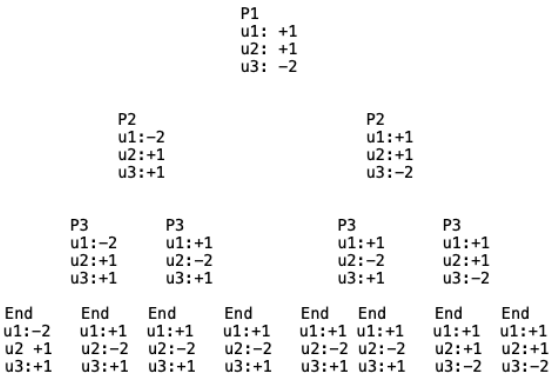

Fig. 5: Each player maximize own utility in the search tree. This example every player has corresponding two choice.

Algorithm 1 Multimax

if *booLose* is true **then**
  loser utility = -2, winner utilty = +1
else
  Find the best utility of each turn in the subbranchs using multimax recursively
end if

IV. Machine learning model

The selected machine learning model is Q-Learning. Q-Learning is an important form of reinforcement learning. And it use Q-value to update every step of iteration. The algorithm flow of Q-learning is as follow:

1. Initialize Q-values arbitrarily for all state-action pair

For our experiments, we just set initialized Q-values to all zeros.

2. For life or until learning is stopped.. For our experiments, we will stop until two players have no space to move. And the remaining player is the winner.

3. Choose an action(a) in the current world state(s) to maximize the current Q-value estimates.

4. Take the action (a) and observe the outcome state (S') and reward (r).

5. Update Q(S,A) := (1-α)$Q(S, A) + \alpha(R + \gamma Q(S', A'))$

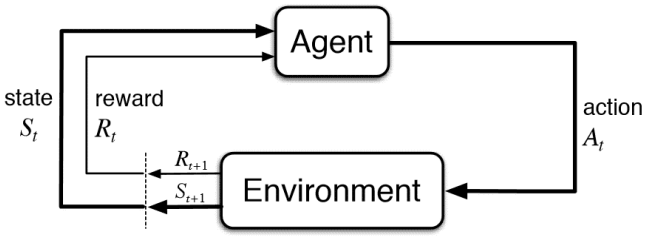

Fig. 6: Q-Learning

V. Baseline opponent

The baseline opponent is adopted to achieve the goal of completing and evaluating the performance of the AI. For comparing that baseline without any heuristic knowledge, we generate the the randomness decision player. That means every legal step has equally probability to be selected. For instance, there are just 2 step with each 50 percent to be moved with in the Figure. We design 2 randomness palyer as baseline opponents to compete and compare with AI generated by Multimax and Q-Learning Algorithm.

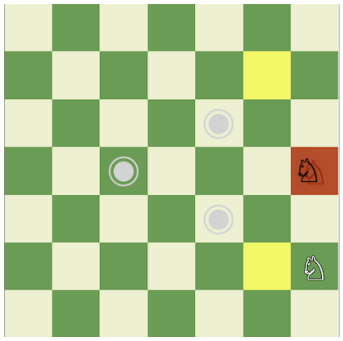

Fig. 7: Q-Learning

VI. Experimental results

A. Individual Task

In this experiment, Lu Han designed and implemented the Multimax.

Runxin Peng and Yan Gong designed and implemented two versions of Q-learning. First method is Mainly implemented by Yan Gong, supported by Runxing Peng. Second method is Mainly implemented by Runxing Peng, supported by Yan Gong.

Yankai Jin helped analysis for Q-Learning.

B. Multimax

There are 5 configuration to test the feasibility and reability of the Multimax algorithm: [3x4],[4x3],[4x4],[4x5],[5x4] respectively. We randomly initilize three players with 100 games using each cofiguration In the final result of Multimax, the AI player with multimax algorithm exceeds the average

randomness player as Show in Figure. It shows that the performance of AI is better especially in large scale chess.At the same time, the efficiency decreases rapidly as the scale enlarges as shown in the Figure.

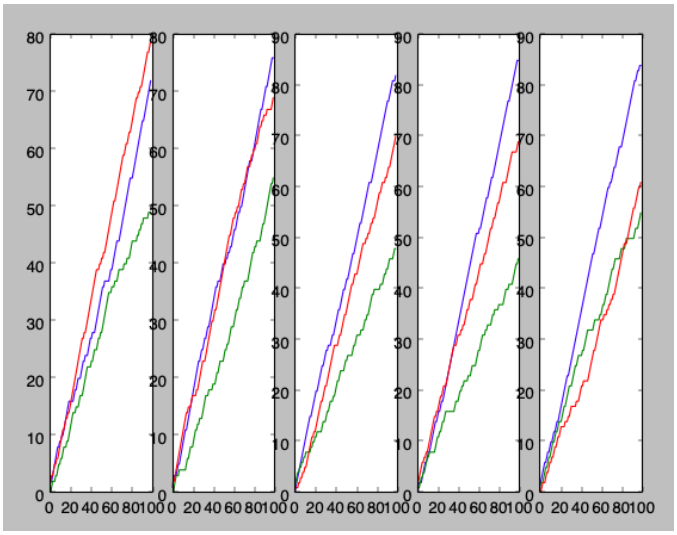

Fig. 8: The 5 configuration([3,4],[4,3],[4,4],[4,5],[5,4]) of Multimax win rate graph.
P1(Red): AI Player, P2(Blue):Randomness player, P3(Green): Randomness player

	P1 Artificial Intelligence Win Rate	P2 Randomness Win Rate	P3 Randomness Win Rate
3 x 4	72	49	79
4 x 3	76	55	69
4 x 4	82	48	70
4 x 5	85	46	70
5 x 4	84	55	61

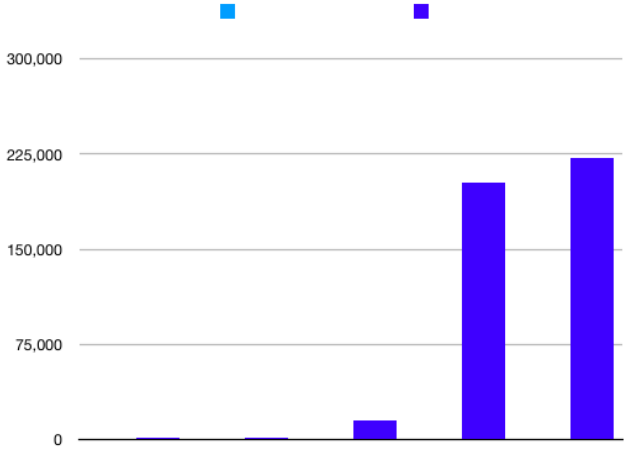

Fig. 9: The 5 configuration([3,4],[4,3],[4,4],[4,5],[5,4]) (left to right) of Multimax number of node checked.
P1(Red): AI Player, P2(Blue):Randomness player, P3(Green): Randomness player

	3 x 4	4x3	4 x 4	4 x 5	5 x 4
nodes checked	1092	983	15172	202455	222423

C. Q-Learning

We maintain a Q-table which will record the long-term reword for different state and action. There are 8 actions in out game. And we use two different ways to record state. Also, we use a variable epsilon to decide whether to explore or exploit when training.Then, we will use Bellman Equation to update Q(s,a).

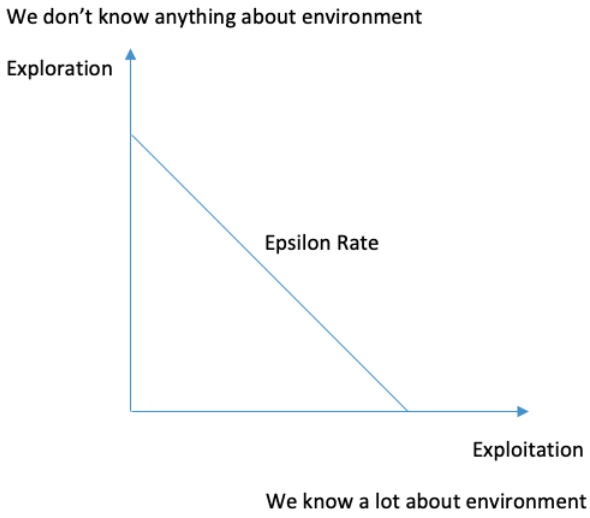

Fig. 10: Exploration vs Exploitation

First way (Mainly implemented by Yan Gong, supported by Runxing Peng):

There are three players, play on a 5*5 board. And we take the current player position and accessed position on board into consideration when creating the state. Besides, there are 8 actions, and we need to maintain a Q-Table, which is consist of states and actions. As showed in the following fig, with increasing of the iteration. Our Q-Learning is achieving better and better win rate. Here is the win rate in 10000 games with different number of iterations during Q-Learning training

Algorithms	Win rate
Play with Random	0.6907
Q-Learning with 10 iterations	0.7040
Q-Learning with 100 iterations	0.7188
Q-Learning with 1000 iterations	0.7314

Fig. 11: Win Rate

Second way (Mainly implemented by Runxing Peng, supported by Yan Gong):

There are two players, play on a 7*7 board. And we only take the two players position into consideration when creating state. Besides, there are 8 actions, and we need to maintain a Q-Table, which is consist of states and actions. As showed in the following fig, with increasing of the iteration. Our Q-Learning is achieving better and better win rate. Here is the win rate with different number of iterations during Q-Learning training

Algorithms	Win rate
Q-Learning with 200 iterations	0.62
Q-Learning with 1000 iterations	0.65
Q-Learning with 2000 iterations	0.68
Q-Learning with 10000 iterations	0.7

Fig. 12: Win Rate

VII. Conclusion

We implemented the Artificial Intelligence of gamer by Multimax and Q-learning. The most significant result is the performance of two models of AI. The AI performs best in large scale, especially 4 x 5 and 5 x 4.And from the Q-Learning, We can see the win rate increasing gradually with the number of iteration and finally convergence to a certain level of win rate. The most challenging part of this project is simplified the model. Although the concept of the game is easy, it still has multiple fields to be simplified. We can only make reasonable decision overcome that. If we were going on, we would strengthen efficiency of the algorithm, for example using alpha-beta pruning. We would also compare this two algorithm with traditional supervised learning.

References

[1] (). Knight-isolation, [Online]. Available: https://knight-isolation.herokuapp.com.

[2] (). Knight-isolation, [Online]. Available: https://github.com/dingran/knight-isolation.

[3] W. Pijls and A. de Bruin, "Game tree algorithms and solution trees," *Theoretical computer science*, vol. 252, no. 1-2, pp. 197–215, 2001.