Chu Thomas – S2B1
Guillou Aurélien – S2B1
Giraud Thomas – S2B1
Hernandez Lilian – S2B1
Nguyen Simon – S2C1
Ba Sidi – S2A1

IUT de Belfort
Département Informatique
Groupe 10

Constructing two Kamisado playing agent

COMPARATIVE STUDY

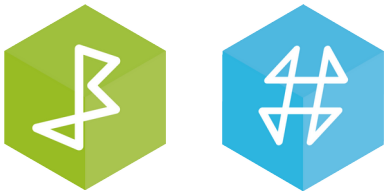


Table of Contents

1 About Kamisado

Kamisado is an abstract board game for 2 players, played on a grid with differently colored squares. Each player has 8 pawns, each one corresponding to a tile color. The first player, black, can choose which piece to move. After the first move you must always move your piece that corresponds to the color of the square the last player moved to. You can move a piece as far as you want straight forwards, or diagonally forwards as long as you don't encounter the opponent's pawn. Never sideways or backwards. You win by moving one of your pieces to the opponent's home row.

2 AI algorithm description

We created two AI for this Kamisado game:

- a said "naive" AI
- an AI using the minimax algorithm

The purpose of this report revolves around the question of how to construct a Kamisado AI that holds the power of winning, hopefully. Our approach is quite frank, the first AI was fully developed by our logic and the second one is heavily inspired by the game tree logic using the minimax algorithm that is described in the following sections.

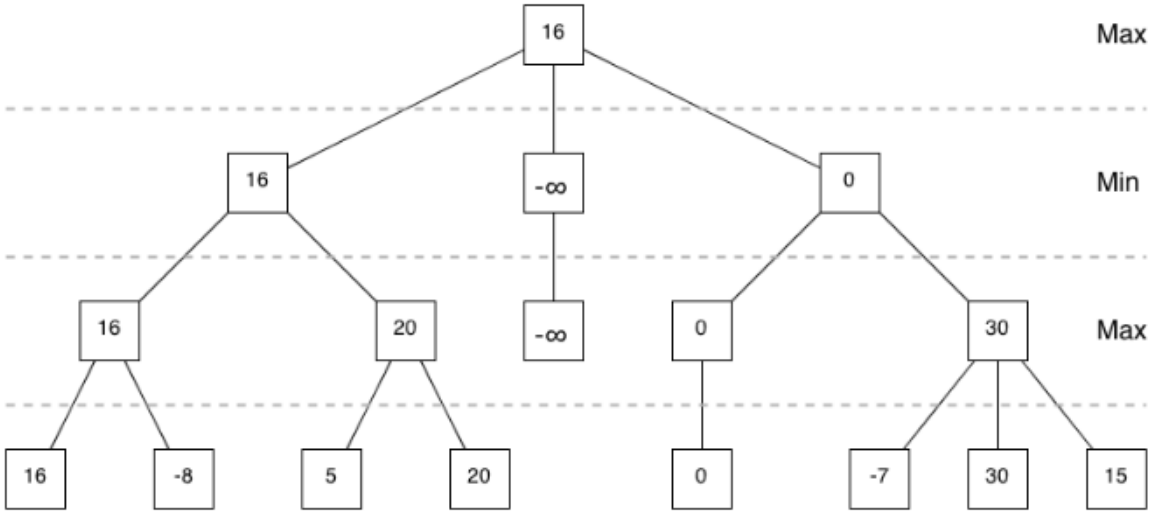
2.1 Naive algorithm

The concept of the naive AI is fairly simple, we use two for loops to go through every 64 cells and locate every pawn of the AI. The color of the last tile the opponent moved to is given to us and the matching pawn is selected. A list called *reachable* is created containing every available coordinate the selected pawn can go to. If this list is empty, the player cannot move, therefore the turn is given to the opponent. If the list is not empty, we determine if there's a direct winning move, if there is one, the pawn goes there. If there isn't, then we use a for loop to go through each coordinate, determine their own *reachable* list for each one to find if there is a possibility of winning the next turn. If this possibility exits, the coordinate will be the next move, if it doesn't exits, then we'll pick a random element in the former *reachable* list.

Pretty simple yet fairly effective, the logic of the AI is centered on the possibility of winning looking at most at 1 turn in the future.

2.2 Minimax algorithm

Things are now getting more interesting with the minimax algorithm. It searches the game tree of a Kamisado game and return the best move to make from a given position. The game tree of a game is a tree structure in which the nodes are possible states of the game and edges represent legal moves connecting different states. Looking at the game tree, one can try to analyze what will happen in the future if the opponent selects a particular move :



The algorithm does a search revolving around a *depth* variable, this variable is essentially how many turn do we observe in the future. In the example above, a node that has the value 0 means a draw, -∞ and +∞ meaning both a victory. To understand better the tree, the best move to make is the move to the child of the root node that has been assigned the highest value, the worst move being the lowest value. The value of each node is determined by an heuristic evaluation, this value is an evaluation of the “possibility” that the game state will be one by one of the players.

This is way more complicated than the former AI but has the merit of being more interesting and closer to what a real board game AI looks like.

3 Comparison

We will now do some small performances comparison between both AI:

- Counting how many times we made elementary operation in both AI, we can evaluate the “weight” of the algorithm on the computer. In the minimax algorithm, there was 7,547,335 elementary operation for one move while the naive AI only needed 71 at most for 1 move meaning the minimax algorithm is way more heavy and need more data to execute.
- We also tried to measure the elapsed time, find the execution time. Using a profiler, we estimated an execution time of 11.534336ms elapsed time for the naive AI and 1795.16211ms elapsed time for the minimax AI, leaving a clear difference. Not a surprising conclusion considering the length of the minimax AI and also its recursive nature.

4 Conclusion

Ultimately, while the minimax AI is quite heavy with a concerning time complexity, this AI does way better against a human than the naive AI. Although one big problem of the minimax AI is that it does really well against humans but not so well against similar agents or a human who knows how to behave against that particular AI. It does make for a decent agent against amateur Kamisado player but get easily defeated by experienced player.