CMPT 310 - Connect 4 Project Report

Qingzhuoran Guo

qga6@sfu.ca

301265603

Zhang Yan

zya49@sfu.ca

301300258

Environment and Organzation

This project is made based on C++ with standard libraries. The program runs in basic Linux system terminal. Instruction on how to run the program and game rules are included in "README" file.

Algorithm

1) Base algorithm

The first part of the project uses pure Monte Carlo Tree Search (pMCTS), which is: for each time the AI wants to make a move, it makes a certain number of random playouts. For each random playouts, it takes actions from the set of all possible legal moves. For each of those legal moves, it simulates the game after of making that move – with all choices randomly generated from the set of legal moves. Once the simulated game finishes, record the result (win, lose or draw). Once all the simulate games are finished, calculate the probability of winning for each of the legal actions, and choose the move with highest probability of wins.

2) Modified algorithm

The modified version of the program is still based on pMCTS with a very 'light' Depth-Limited Search (DLS). In the simulated game mentioned described in part 1), the modified AI will not blindly generate the next move by random but do a 'search' first: for each $x \in$ {legal moves}:

1. Check if x wins the game. O(1)
2. Check if the opponent can win the game with his next move $y \in$ {legal moves with x placed}. O(7)
3. Check if x is a **Good Move** (GM, which means with the current GM placed, the player **can** win the game with his next move no matter where the opponent place their move). O(7^2)
4. Check if the opponent **has** a Good Move with x placed. O(7^3)

If any of the above is true (in order), it means that the AI can make the decision of whether the current move x is a 'win move' or 'lose move' immediately. Then it can skip the random playouts part as part 1) mentioned and return the win rate of x. However, if none of the 4 checks are true, it means it's still 'debatable' whether x is a good move, therefore the AI will go back to random playouts algorithm as what part 1) described.

3) Analysis (theoretical)

It is obvious that the AI based on the base algorithm will not be a good player because the algorithm has problems:

1. The algorithm is like greedy algorithm, it only cares about the present state; it will only generate probabilities for the current legal moves and it does not care about move sequences. For example, it is possible that for x, $y \in$ {legal moves} and x has 60% of win rate and y has 55% of win rate, but the probability of placing x right after y may be 100% (y is GM) while placing y after x only has 50%. The base algorithm does not consider any of these situations.

2. The algorithm is not time efficient. Consider case 1 and 2 mentioned in part 2), in those situations, more playouts will not improve the results:
    a. In case 1, the probability will always stay as 100%; no need to do more playouts.
    b. In case 2, a 'smart' opponent will make the win-move, there's no need to spend time calculating that.

Based on those problems, the modified algorithm is ideally better than the base version because:

1. Although the algorithm is still "short sighted" – it only cares about two moves from the current states, it should still prevent a "smart opponent" from making GMs from the current state – the biggest losing issue of the base version. And the chances of making GMs of GMs of GMs of GMs … (until the win) is much more difficult and sometimes impossible – may require complete Depth First Search which each iteration takes time O(7^(24-turns)). Therefore, AI based on this algorithm can ideally be better than normal "smart players" who can only think of a few steps ahead.
2. By checking "immediate win/lose", the algorithm saves time from unnecessary playout calculations. The extra cost of checking those 4 conditions takes O(400), which saves a lot of time when one of the conditions are true, and since it will only check those conditions once, it should not have huge impact on the running time.

Implementation

Limits. To make sure a pleasant experience for users, a time limit (TIME_LIMIT) is set to be 2 seconds, which forces the AI to make its choice within 2 seconds. In addition, to maximize the accuracy of probability calculated from each simulate game, the limit of number of playouts (PLAYOUTS) is set to be 100,000.

Multi-threading. Because of the complexity of calculation (especially with modified version), it is hard to get an accurate, correct and time efficient solution in a single thread. Therefore, the AI decision making handler (AI_decision ()) creates 7 child threads and make each of them responsible of calculating 1 possible move. With multi-threading, the program can now handle at least 45,000 (up to 100,000) playouts for each possible move within 2 seconds.

Developer mode. Aside from the "user mode", a developer mode is implemented to help display some important information such as states of each thread, development of probability calculation and GM-finding states. The developer mode is turned on by default to show how the data in the next section are collected, it can also be turned off easily (follow instructions in README).

Versions. The AI decision maker (AI_decision ()) has an "version" parameter which indicate which version of the algorithm it will use. 1 represents the base version, and 2 represents the modified version.

Tests and analysis

1) Correctness/performance (limited tests)

An online version of Connect 4 is used to play against with our AI: https://www.mathsisfun.com/games/connect4.html, difficulty is set to hardest. And Aaron Guo participated as the human player (entry level Connect 4 player)

Base algorithm:

vs. online bots:

	Game1	Game2	Game3	Game4	Game5	Game6	Game7	Game8	Game9
first	lose	lose	lose	lose	lose	lose	lose	lose	lose
second	lose	lose	lose	lose	lose	lose	lose	lose	lose

Go first-win rate: 0%

Go second-win rate: 0%

vs. human (Aaron Guo):

	Game1	Game2	Game3	Game4	Game5	Game6	Game7	Game8	Game9
first	draw	win	draw	draw	win	draw	draw	draw	win
second	draw	win	win	draw	draw	draw	win	draw	draw
second	lose	lose	lose	lose	lose	lose	lose	lose	lose

Go first-win rate: 33.3%

Go second-win rate: 33.3%

Modified algorithm:

vs. online bots:

	Game1	Game2	Game3	Game4	Game5	Game6	Game7	Game8	Game9
first	win	lose	draw	win	draw	lose	win	lose	win
second	win	lose	win	lose	lose	lose	lose	win	lose

Go first-win rate: 44.4 %

Go second-win rate: 33.3%

vs. human (Aaron Guo):

	Game1	Game2	Game3	Game4	Game5	Game6	Game7	Game8	Game9
first	win	win	win	win	win	win	win	win	win
second	draw	win	win	win	win	draw	win	win	win

Go first-win rate: 100%

Go second-win rate: 77.8%

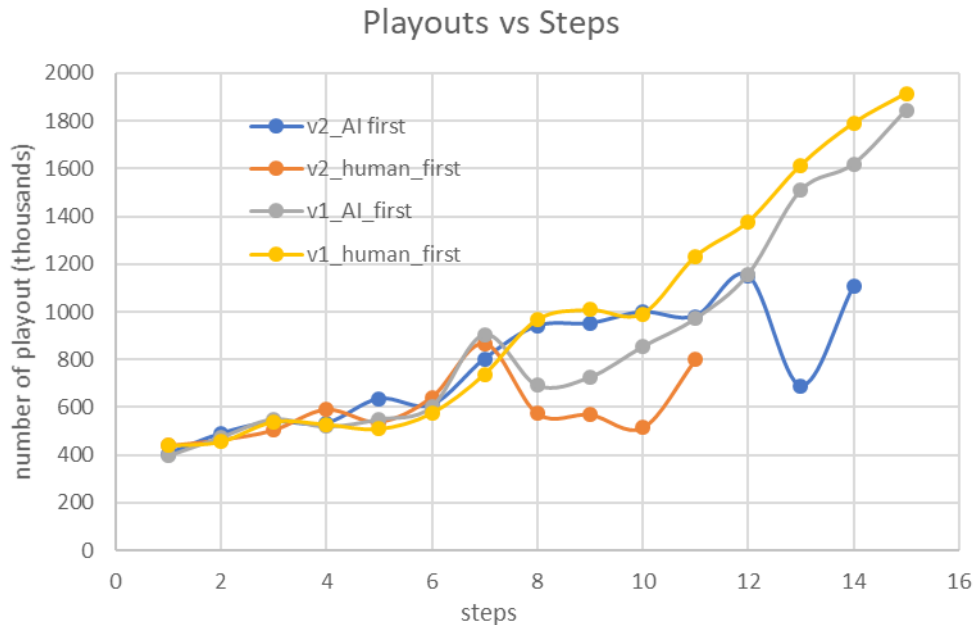
vs. Base algorithm:

	Game1	Game2	Game3	Game4	Game5	Game6	Game7	Game8	Game9
first	win	win	win	win	win	win	win	win	win
second	win	win	win	win	win	lose	win	win	win

As shown in the tables above, the results show that neither the online game nor the AI's are perfect: they do not have a 100% first-win rate. And the online version of Connect 4 is very similar with the AI implemented in terms of win rate.

For AI based on the base algorithm, it lost all the games and does not have a good win rate play against human at an entries level. On the other hand, the modified algorithm does a much better job: 1. It has a much higher win rate against the online bot and even 100% first-win rate and 77.8%-win rate against human (and never loses).

2) Time complexity in terms of playouts calculated (a typical test result)



As shown in the graph above, both versions of algorithm (v1: base, v2: modified) share a similar trend of number of playouts until at some point (for example at step 7 and 12), AI-version2 's number of playouts suddenly drop (4-conditions met, decision made at once) and find the "win move" (at step 11 and 14). On the other hand, the graphs of AI-version1 are relatively stable as expected (increase linearly).

This typical graph supports the algorithm analysis mentioned above: the modified algorithm should perform generally better than the base version in terms of correctness (find the win path) and time efficiency.

3) Other observations

1. One interesting fact is when the number of playouts of modified version drops, it likely means the program finds a "win move" instead of "lose move" which should theoretically have the same probability. This may due to too few tests cases are made.

2. "Thinking 2 steps ahead" (modified version) is good enough to beat most of the online small Connect 4 mini games (with approximately 50% win rate and 20 % lose rate). And this may also due to too few test cases are made.

Discussion

Limitations

Because the test phase requires players to manually input the moves and record the data, very limited number of tests are made. Therefore, although the test results support the algorithm analysis, the result is not strong.

Possible improvement

The modified algorithm uses a light Depth-Limited Search as heuristic to skip random playouts in some cases, this may be improved in several ways such as creating pattern databases: we can take the states where the algorithm find a GM and save the state to the database, and use the database as heuristic.

Conclusion

Although the modified algorithm (2-steps ahead thinking) is not perfect as it will not guarantee a win when it makes the first move against some good algorithms, it is still a huge improvement compare to the base version in terms of correctness and time efficiency.