2204939: Adversarial Search Evaluation

For the game of Connect, multiple variables directly affect the performance of the Minimax algorithm and its alpha-beta variation. These variables include (and not limited to) the board size of the game, the number of pieces in a line required to win, the depth used for the algorithm, etc... The variables mentioned above were what were used in the evaluation of the performance of the minimax and minimax with alpha-beta pruning algorithms implemented. For each variable measured, it was measured against the randomPlayer opponent and an intelligent opponent (a player using alpha-beta minimax with depth 3).

1 Evaluation of Minimax without pruning

1.1 Game Size

To measure how the board size and the number of pieces in a line required to win affects the algorithm's performance, a variable called "Game Size" is used. This simply refers to a scale of the game where a game could be : **small:** a board size of (3,3) and required pieces in a line to win is 2. **medium:** a board size of (6,7) and required pieces in a line to win is 4, **large:** a board size of (9,10) and required pieces in a line to win is 7. *where each (x,y) has x for the number of rows and y as the number of columns.* The results of the algorithm's performance (The number of nodes expanded by the algorithm and time taken to complete a game) against the randomPlayer opponent are shown in Table 1.[1]

Board Size	Nodes Expanded	Standard Deviation	Time taken for a game (s)	Standard Deviation
Small	116	5.6	0.019601	0.038
Medium	3,963,614	965,808	418.2	114.3
Large	100,078,600	29,015,220	10020.6	3291.9

Table 1: Impact of Game Size on performance against an opponent choosing random moves.

As the game size grows, the number of Nodes Expanded and Time Taken increases exponentially[2]. The win rate for each game size is as follows: **small:** 100%, **medium:** 98.4%, **large:** 97.3% The same test was done against the intelligent opponent. The results are shown in Table 2.[3]

Board Size	Nodes Expanded	Time taken for a game (s)	Standard Deviation
Small	123	0.013	0.035
Medium	4,055,518	422.5	24
Large	92,563,286	7996.4	2953.8

Table 2: Impact of Game Size on performance against an intelligent opponent.

It is clear to see that the performance against a random opponent and an intelligent opponent is very similar. However, against an intelligent opponent, the algorithm wins for a **small** game, loses for a **medium**-sized game, and draws for a **large** game.

[1] 550 test cases were used to collect this data, values for Nodes Expanded and Time Taken is an average across those 550 tests

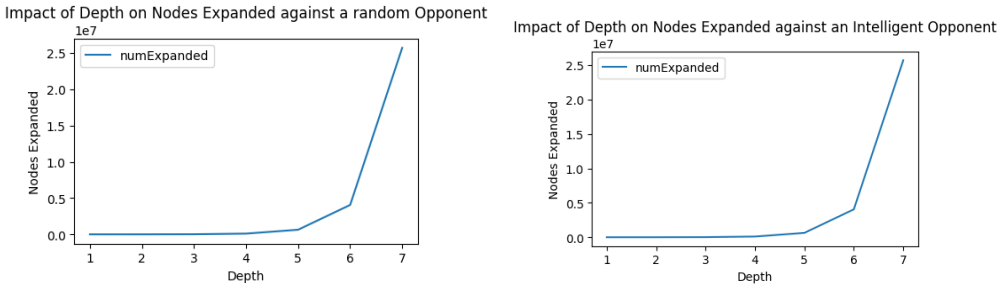
[2] since as the game size increases, a random opponent can play a game in more ways, so across multiple tests, the gameplay can vary differently so the standard deviation increases massively

[3] Since minimax will play against minimax with alpha-beta pruning, any game played for a given game size will have the exact same moves in succession, so no variation in Nodes Expanded

1.2 Depth

The impact of the algorithm's depth on the algorithm's performance was also measured. Results are shown in figure. 1.

Figure 1: Depth effect on number of nodes expanded.



From these two graphs, we can see that the average number of nodes expanded[4] has an exponential increase as the depth of the algorithm increases. This is the case against both a random opponent and an intelligent opponent. The time taken to complete a game exhibits the same behavior since there is a proportional relationship between Nodes Expanded and Time Taken to get a move.

2 Evaluation of Minimax with alpha-beta pruning

In alpha-beta variation of minimax, optimizations were made. It starts exploring from the middle column outward instead of left to right. The evaluation function prioritizes the most immediate path to a win by returning a value linked to the depth when called.

2.1 Game Size

The same test that was used in section 1.1 was done for minimax with Alpha-Beta pruning. The results are shown in Table 3 and Table 4.[5]

Board Size	numExpanded	std	numPruned	std	Time taken for a game (s)	std
Small	34.6	0.9	8	0	0.01	0.027
Medium	44128.9	23,021.3	14650.2	9,260.1	5.9	3.4
Large	318,399.5	157,683.3	111,275.1	62,851.4	45.0	25.7

Table 3: Impact of Game Size on performance against an opponent choosing random moves.

Board Size	numExpanded	numPruned	Time taken for a game (s)	std
Small	36	8	0.012	0.032
Medium	52,115	15,556.	7.4	1.66
Large	389,394	140,382	55.8	10.2

Table 4: Impact of Game Size on Performance against an intelligent opponent

[4] 550 test cases were used to get the number of nodes expanded against a random opponent

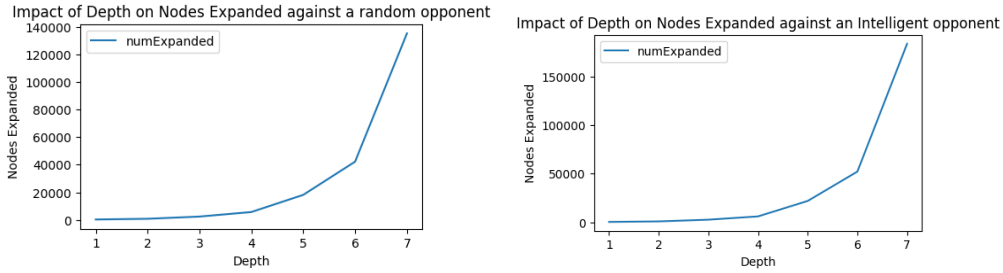
[5] To fit all data in the table. Average Nodes Expanded is shortened to numExpanded, Average Nodes Pruned is shortened to numPruned, and Standard Deviation is shortened to std

For a random opponent, the algorithm's win rate is as follows: **small** game: 100%, **medium** game: 100%, **large** game: 99.6%. So it is clear to see that the optimized alpha-beta version of the algorithm outperforms a standard minimax algorithm against a random opponent. However, there is a exponential growth in time taken and numExpanded as the game size increases. It is clear to see that the performance against a random opponent and an intelligent opponent is very similar. However, against an intelligent opponent, the algorithm wins for a **small** game, wins for a **medium**-sized game, and draws for a **large** game.

2.2 Depth

The same test for measuring the effects in performance in section 1.2 is done for the alpha-beta pruning variation of minimax The results are shown in Figure 2.

Figure 2: Depth effect on number of nodes expanded.



Again, just like section 1.2 there is an exponential increase in the number of nodes expanded as the depth increases (and therefore an exponential increase in time taken to complete a game as depth increases).

3 Conclusions

It is clear to see that the general behaviour of both algorithms are the same as Game Size or Depth changes. For both variables, if they increase, the number of nodes expanded and time taken to complete a game increases exponentially for both algorithms. However, the magnitude of these two variable are orders of magnitude larger in the standard minimax algorithm; the value for nodes expanded for a large game for minimax is 92 million whilst for the Alpha-Beta variant is only 318,000 ! Therefore, this shows how much more efficient the alpha-beta pruning version is to minimax, since there are about 111,000 branches pruned, therefore saving a large amount of time by not exploring unncessary nodes.
The same behaviour is displayed for testing how the depth of the algorithm affects the number of nodes expanded, where both algorithms showed an exponential growth in nodes expanded as depth increases, whilst minimax being order of magnitude larger than minimax with alpha-beta pruning; with about 26 million nodes expanded for minimax and about 180,000 nodes expanded for minimax with alpha-beta pruning (both running at depth 7) where minimax with alpha-beta pruning around 50,000 branches.