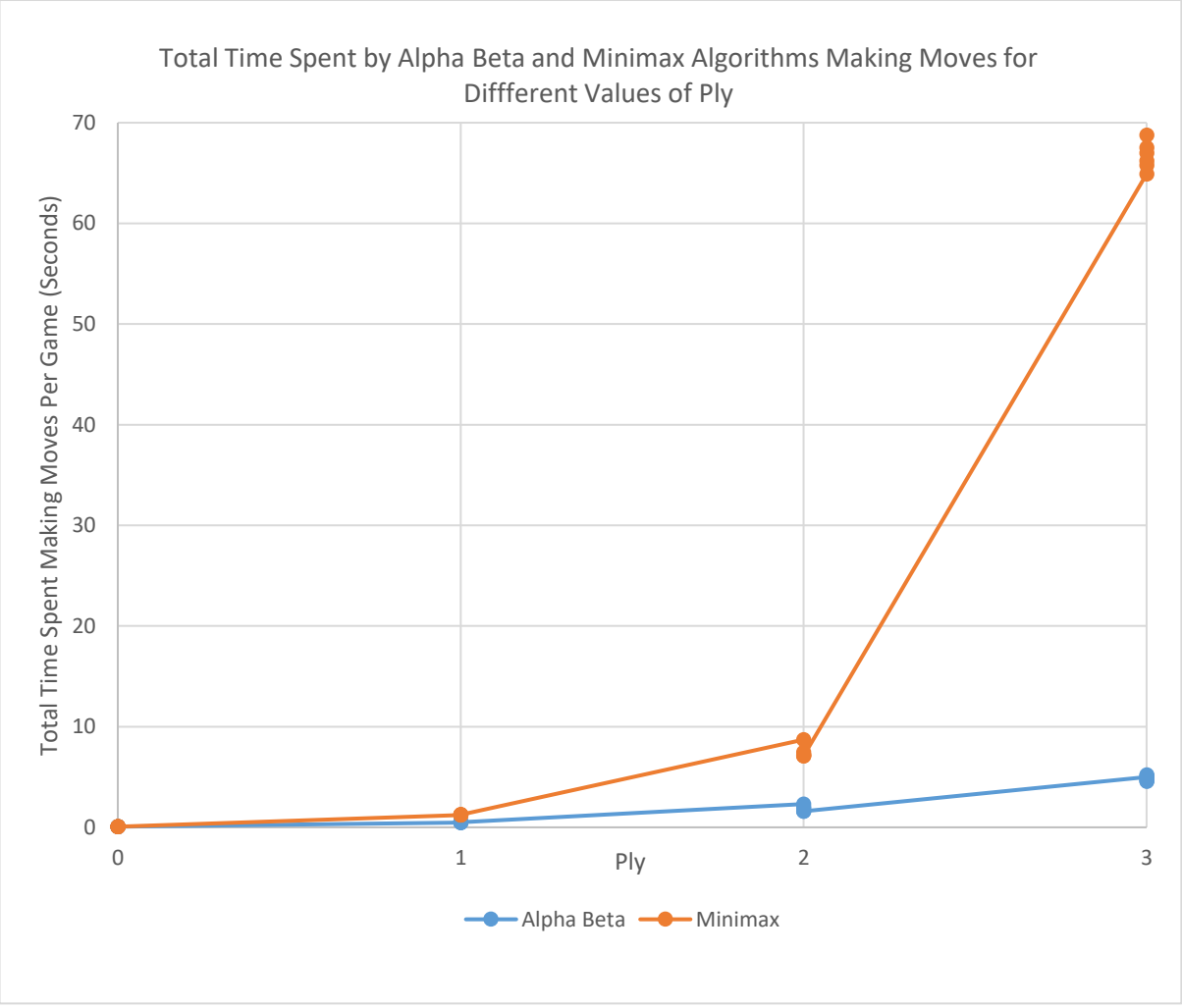
The Effects of Alpha-Beta Pruning on Minimax Search in Connect 4

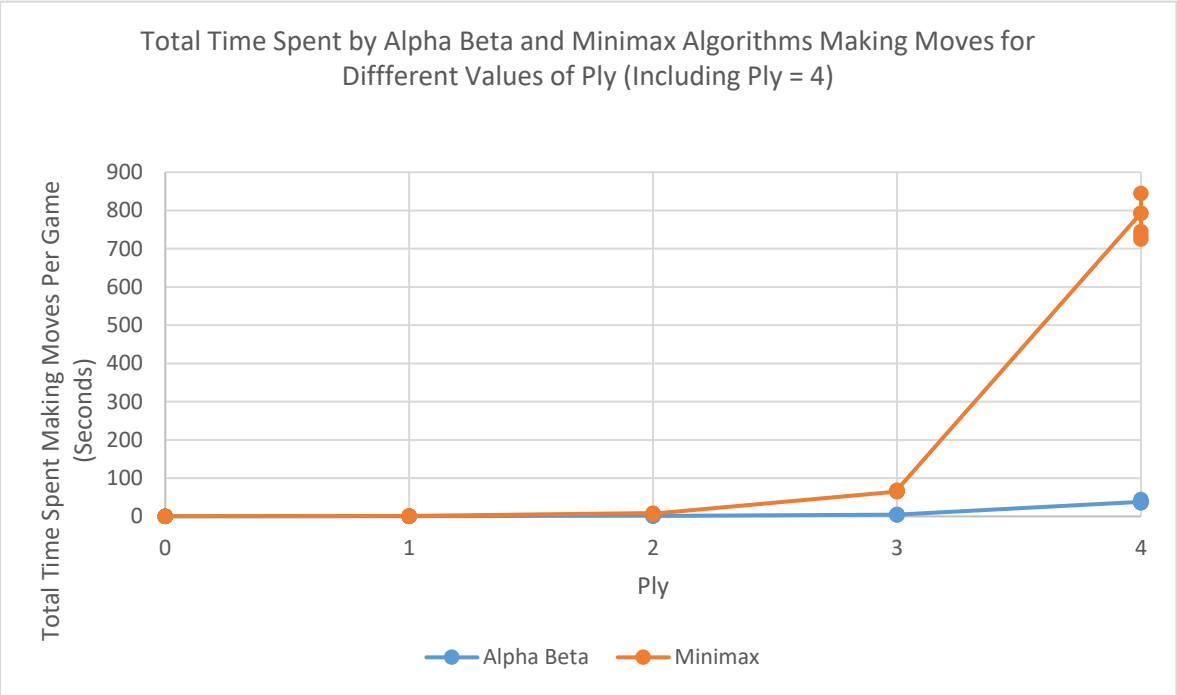
Data Collection

In order to collect data to examine the effects of Alpha-Beta Pruning on the pure Minimax search I played the Minimax AI against the Alpha-Beta modified AI on the game of connect 4 for different values of ply. For each ply value I played the two AIs against each other 6 times, and for each game recorded the total time spent by each AI making moves in the game, the mean time spent by each AI per move in the game, the times spent by each AI per move in the game, and the winner of the game. In each game the AI assigned red tokens played first, so to account for any advantage/disadvantage to be had from being the first to play I let each AI be the red player 3 times for each value of ply. The range of values I set for the ply were 0-4 inclusive, as the games with a ply of 4 each took over 10 minutes, so higher values of ply would have taken too long due to the computational restrictions imposed by my laptop. After collecting the data I then plotted a scatter graph, with each point representing the total time spent by one of the AIs making moves during a game with a given ply, enabling me to clearly see the performance improvements that Alpha-Beta pruning caused. I also created a table showing the mean time taken per move by each AI for different values of ply and a table showing which colour won each game for different values of ply.

Scatter Graphs



The Effects of Alpha-Beta Pruning on Minimax Search in Connect 4



Tables of Results

Ply	0	1	2	3	4
Mean Time Spent by Alpha-Beta per Move (seconds)	0.00692	0.01243	0.09474	0.12181	0.97349
Mean Time Spent by Minimax per Move (seconds)	0.00668	0.03084	0.38205	1.66738	19.31999

Ply	0	1	2	3	4
Winner	Red	Blue	Blue	Red	Blue

Performance Analysis

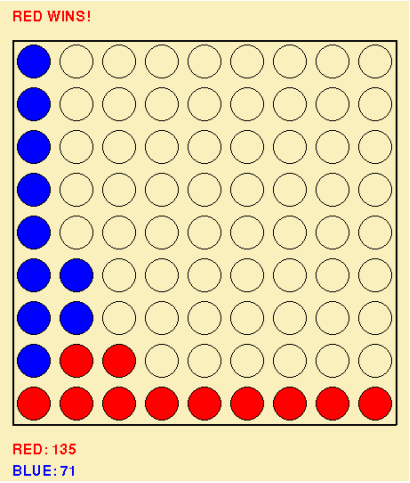
Looking at the graphs showing the total time spent by the two AIs making moves per game and the table showing the mean time spent by the two AIs per move it is clear that the Alpha-Beta AI is much more efficient at finding the optimal move for different values of ply. For smaller values of ply, this difference isn't as large as for bigger values of ply. For example, with a ply of 1, the mean time spent making each move by the Alpha-Beta AI is 40.3% of the mean time that the Minimax AI spends taking each move. Although this is a significant improvement, when the ply is 4 this improvement is

even more significant, with the mean time spent making each move by the Alpha-Beta AI being 5% of the mean time that the Minimax AI spent taking each move. This reveals just how much more efficient the AI is which implements Alpha-Beta pruning, as it will still find the same optimal move as the Minimax AI but in a fraction of the time. It also illustrates how as the depth of the search tree is increased, the benefits gained from optimizing the Minimax Algorithm with Alpha-Beta pruning become increasingly larger in terms of time taken to evaluate the search tree.

The win-ratio of the Alpha-Beta AI was 50%, as during the tests the two AIs played as the 1st/2nd player 50% of the time. Although Alpha-Beta pruning reduces the time spent by the AI finding the optimal move for a given ply, the optimal move it returns is still exactly the same as a Minimax AI searching to the same ply. In other words, Alpha-Beta pruning only reduces the time taken to find the optimal move, as it rules out sub-optimal moves more quickly than pure Minimax search, but it doesn't change the optimal move found; the optimal move found using Alpha-Beta pruning is the same as that found by the pure Minimax algorithm. Therefore, if the Alpha-Beta AI is playing against the Minimax AI searching to the same depth of the search tree, the winner of the match is dependent on who plays 1st/2nd, as they both return the same optimal moves as each other. This was interesting as it illustrated that for 3 out of the 5 values of ply I investigated, it was advantageous to play 2nd (blue) rather than 1st (red).

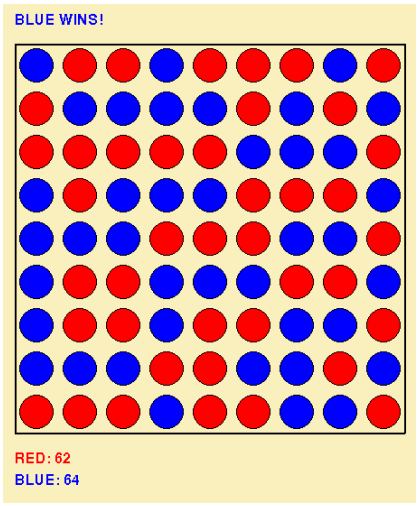
However, although both algorithms return the same optimal moves, the difference in their time taken to return these optimal moves would cause me to choose the Alpha-Beta AI in a tournament. This is because in a tournament I would want to use as high a value of ply as possible (although it wouldn't need to exceed 81 as there are only 81 places on the board so the AI wouldn't need to play out over 81 moves into the future), as using a higher value of ply enables the AI to look a greater number of moves into the future, enabling it to choose better moves based on longer-sighted predictions of how the game will be played if the opponent plays optimally. Using the Alpha-Beta AI, therefore, would vastly reduce the time spent by the AI in finding the optimal move when using a large value for the ply. In contrast, the pure Minimax AI would take too long to find the optimal move to play in a tournament, as for large values of ply it becomes increasingly less efficient than the Alpha-Beta AI. Tournaments often impose time limits for moves too, so using the Alpha-Beta AI would enable a greater search depth (ply) than the Minimax AI would achieve in the same time limit. For example, based on the results gathered by running the algorithms on my laptop, if the tournament imposed a time limit on the mean time spent by the AI per move of 1 second, the Minimax AI would have to use a ply of 2 whereas the Alpha-Beta AI would be able to use a ply of up to 4, enabling it to stand a stronger chance of winning its games.

It was interesting to see the optimal playing strategies for different values of ply, so below are some pictures of the final board states for different values of ply:

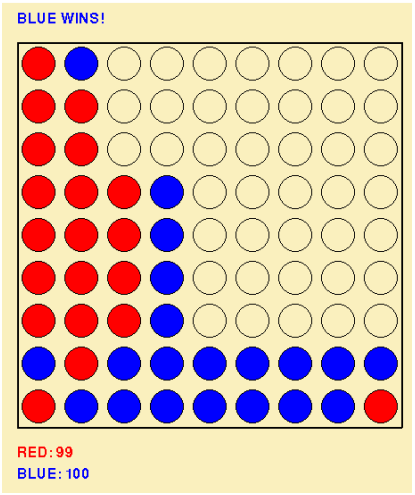


Ply: 0

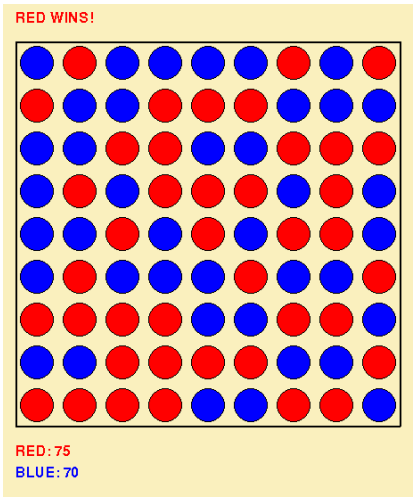
The Effects of Alpha-Beta Pruning on Minimax Search in Connect 4



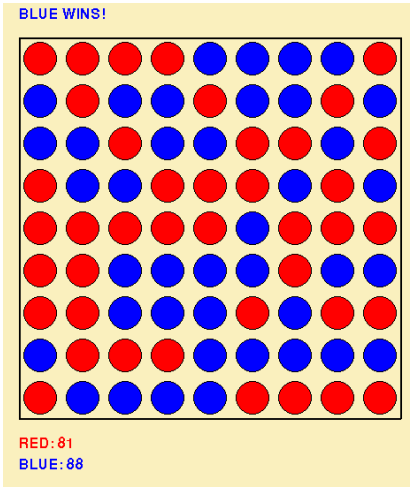
Ply:1



Ply: 2



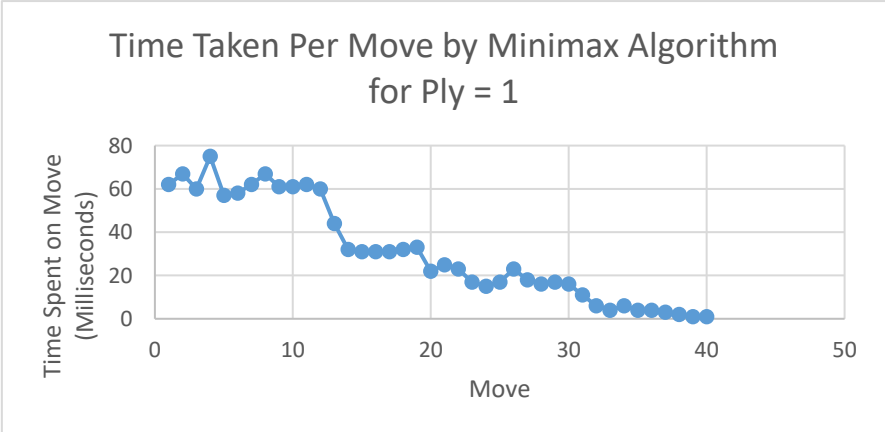
Ply: 3



Ply: 4

Minimax Time Spent Per Move

A final interesting pattern in the data I found was how long the Minimax AI took to make each move, and how this changed throughout the game as the search tree got smaller. Interestingly, by plotting the length taken for each move against the number of the move for different values of ply, the time taken for each move decreases at different rates for different values of ply, as can be seen below.



The Effects of Alpha-Beta Pruning on Minimax Search in Connect 4