Imperial College London

COURSEWORK 2

IMPERIAL COLLEGE LONDON

DEPARTMENT OF COMPUTING

Introduction to Symbolic AI

Author:
Dhruva Gowda Storz, George Yiasemis (CID: 01807283, 01108587)

Date: January 7, 2020

1 Implementation

See comments and documentation in code for details of implementation

2 Difference in search time

The time difference between minimax and minimax with alpha beta pruning is shown in the graph below for a 4*3*3 board.

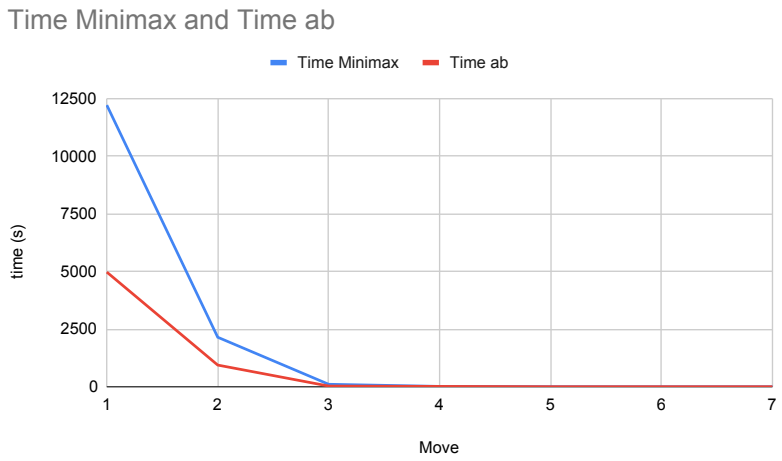

Figure 1: Time taken for search using normal minimax and alpha-beta pruning.

The graph shows that alpha-beta pruning reduces time by a factor of 50% or more for each move. Using a log scale on the vertical axis shows that this reduction in time is consistent over most of the moves, with a few exceptions where it takes the same amount of time as the regular minimax algorithm. However, it is important to note that as the number of spaces remaining for moves goes down over time, the improvement that alpha-beta pruning provides decreases as the number of possible board states decreases.

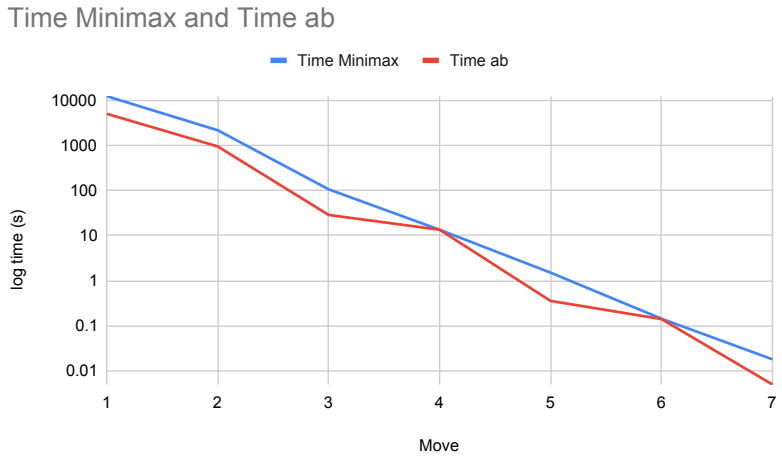

Figure 2: Time taken for search using normal minimax and alpha-beta pruning. y-axis is on a log-scale

We also investigated how the execution time scales up with board size. However, we could not go beyond 4x3 with the resources available to us.

This is due to the fact that the search time depends on a multiple of the factorial of the board area. A 4x3 board is 12! = 479001600 possible board states on the first move, while a 4x4 board is 16! = 2.092279e+13 board states. It would therefore take a maximum of 43680 times as much time as it took to run the minimax algorithm on a 4x3 board, which was unfeasible given our resources.

The search times for various mnk values are shown in the tables below. These search times are averaged over 50 runs, with the exception of the 4*3*3 board, which was only run once due to time constraints.

	2*2*2		2*3*2		2*3*3	
Move	minimax	ab	minimax	ab	minimax	ab
1	0.00091	0.00050	0.0083	0.0030	0.027	0.0098
2	0.00022	0.00022	0.0016	0.0016	0.0047	0.0046
3	0.000084	0.000076		0.00031	0.0010	0.00052
4					0.00028	0.00027
5					0.00010	0.00010
6					0.000083	0.000051
7						
8						
9						

Table 1: time in seconds of calculating moves for mnk 222, 232, 233

	3*2*3		3*3*3		4*3*3	
Move	minimax	ab	minimax	ab	minimax	ab
1	0.029	0.0103	15.26	6.32	12210.60	4969.78
2	0.0054	0.0050	1.68	1.67	2144.41	938.05968
3	0.00094	0.00053	0.20	0.093	105.25	28.49929
4	0.00023	0.00024	0.033	0.026	13.38	13.40128
5	0.000097	0.00010	0.0064	0.0037	1.50	0.35482
6	0.000051	0.000052	0.0014	0.0014	0.14	0.14024
7			0.00049	0.00034	0.018	0.0050
8			0.00021	0.00020		
9			0.000090	0.000087		

Table 2: time in seconds of calculating moves for mnk 323, 333, 433

The tables show that search time increases exponentially with increase in board size (m and n). However, it does not matter whether m or n are increased, as can be seen by the similar times for 3*2*3 and 2*3*3. Increasing k changes the dynamics of the game, which alters the min/max number of moves needed for a win, so it is difficult to comment on its influence on search time. However, the data seems to show that increasing k increases the search time for the first move.
The exponential increase in search time can be seen in the graph below, which plots search time of the first move for each board size on a logarithmic scale.

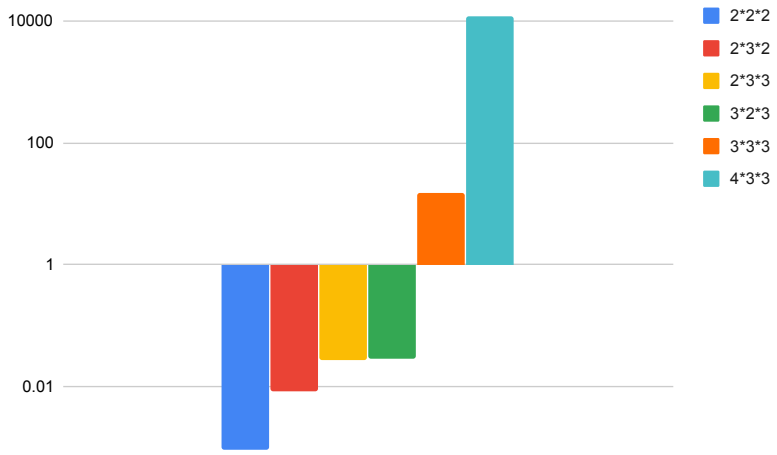

Figure 3: first move search time for boards in ascending size, plotted on a log time scale.