Rohan Ahuja
Daniyal Khan

AI Project 1 Report

Summary:

First, we played the bots against each other, without implementing anything new. The bots implementing minimax (with or without alpha beta) with eval 1 were much better than the random bots, which is expected.

Then, we decided to implement eval 2, which used piece square tables to decide how valuable a piece is on a square. For instance, a knight is more valuable in the center than in a corner, because it has more range. We took Lauri Hartikka's piece square tables and multiplied them by ten, because in Lauri's tutorial the piece material values were lower than ours by a factor of 10. We tested the eval 2 bot against the eval 1 bot, and it seemed to do better for most of the games. One thing we noticed was that during the endgame, both bots would play random moves and not know what to do.

A common strategy mentioned in a lot of popular chess books is "When you don't have a plan, look for pawn moves and pawn breaks". This is how we came up with eval 3, which was to play with passed pawns. A passed pawn is a pawn that remains unchallenged by any opponent pawns from pushing itself towards the 8th rank to promote to a more valuable piece.

Image Source: http://www.chesssteps.com/passed-pawn/

In the position above, c5, a6 and b4 are passed pawns, because they aren't stopped by any of the opponent's pawns. A pawn like f2, for instance, isn't a passed pawn because it can be potentially stopped by g6 and f7 when it pushes forward. Passed pawns are very valuable towards the endgame, when you want to promote your pawn to a queen (most of the time). For eval 3, we needed to make sure our bot still cares about piece values (or else it won't care about giving away important pieces, and only play with its pawns!). So for eval 3, we added the piece square tables to the passed pawn evaluation, to get a combined score - a similar method

to how popular chess engines like Stockfish do it. Just by observing, we noticed that eval 3 was slightly better than eval 2 because it knew how to play with its pawns.

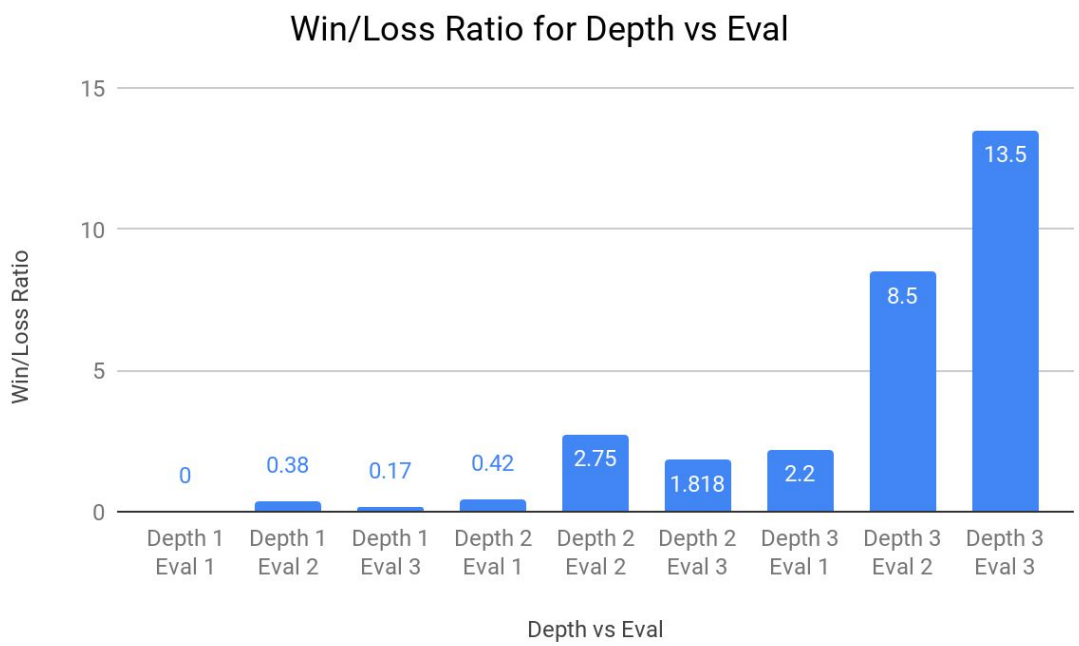
Now that we had our evaluations in place, it was time to test a large number of games. To make our tests quicker, we commented out a lot of console.log commands, and also removed the wait time to make a move. The playGame function is still kept intact, but we added more functions for bot tests. We have a function where we keep the evaluation fixed, but vary the depth. We have another function where we vary the evaluation and keep the depth fixed. We have a function where both evaluation and depth are randomized, and this was the main one used for testing. Also, to see whether depth performs better than an evaluation function, we made a bot with depth 1, evaluation 3 play against a bot with depth 3, evaluation 1. After a large number of trials, we got some interesting results.

Notes:

- To get the bots to play each other, we made two more copies of the Minimax-Alpha-Beta function and just changed the evaluation function call. For the makeMove() function, Algo 4 uses Eval 1, Algo 5 uses Eval 2, and Algo 6 uses Eval 3, but they are all minimax with alpha beta pruning.
- A lot of new functions were implemented, but the main ones used for testing were botTestsRandomized() and botTestsDepthVsEval()
- Eval 2 is evaluateBoardPSQT(), and Eval 3 is evaluateBoardAll()
- To avoid stalemates in winning positions, we added a condition where the bots would return an evaluation value of 0 for stalemates, so it knows it's a draw. This gave us more accurate results.

Discussion:

After playing a little over 400 randomized games, we took the win-loss ratio (wins divided by losses) so we could see which combinations of depth and eval performed better.



Rohan Ahuja
Daniyal Khan

Overall it's clear that Depth matters more, but evaluation matters as well, for example Depth 3 Eval 3 is way superior than the rest. We also notice that Depth 2 Eval 2 did better than Depth 3 Eval 1, which shows us the value of evaluation as well.

Results:

No two identical bots played each other.

Randomized results:
Total games: 424

Depth	Eval	Games	Wins	Draws	Losses	W/L Ratio
1	1	102	0	34	68	0
1	2	100	16	42	42	0.38
1	3	104	10	36	58	0.17
2	1	114	28	20	66	0.42
2	2	80	44	20	16	2.75
2	3	78	40	16	22	1.818
3	1	90	44	26	20	2.2
3	2	98	68	22	8	8.5
3	3	82	54	24	4	13.5

We also played Depth 1 Eval 3 vs Depth 3 Eval 1 for 100 games, and here were the results.

Depth 1 Eval 3 VS Depth 3 Eval 1:

Depth	Eval	Games	Wins	Draws	Losses
1	3	100	0	38	62
3	1	100	62	38	0

Conclusion:

Depth 3 won convincingly, so ultimately depth matters more. But we could look into this further if we had faster chess engines (For example, playing Depth 8 with a strong evaluation function against Depth 15 with a weak evaluation function). There are many optimizations we can use for this, like iterative deepening, null move pruning, transposition tables and so on.