U2265464: Adversarial Search Evaluation

Introduction: - The evaluation of the minimax algorithm on the **Connect 4** game is based upon comparing different metrics such as number of nodes expanded, number of branches pruned, changing different board sizes to check the efficiency of the runtime and fluctuations of win rates by **repeated testing strategies**.

1. EVALUATION OF MINIMAX WITHOUT PRUNING

The first experiment began with comparing different **board sizes** with the **average number of nodes expanded** while performing the minimax algorithm in the search tree. The algorithm starts with a depth of 4 and searches through the tree with **iterative deepening.** The graph illustrates the relationship of the same.

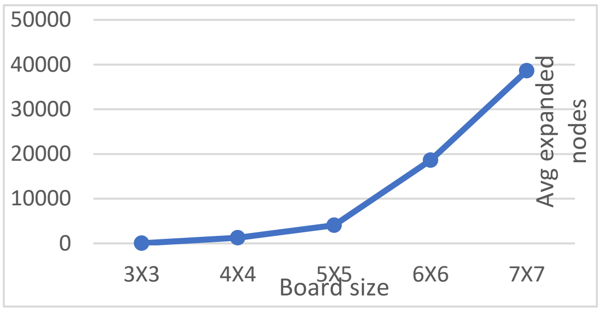

Fig1: Relationship of board size vs avg expanded nodes.

The graph shows a **linear relationship** between the number of nodes expanded per game and different board sizes. Therefore, with an increasing board size, the number of nodes expanded is **directly proportional** to an increasing size of the board.

Other important metrics such as the **average runtime** of the game are also considered and compared with the **board size**. The relationship between the board size and the average runtime is linear until a board of 5X5, and then increases exponentially thus increasing **computation time.** A comparison is also made with the number of pieces in a line against the average number of nodes expanded. The graph is a **logarithmic** whose average number of nodes expanded **increases** with a **decreasing rate.**

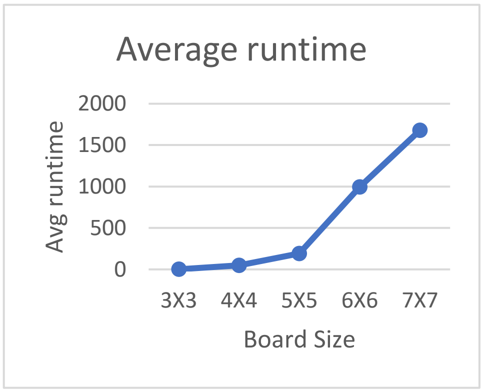

Fig2: Relationship of board size vs avg runtime.

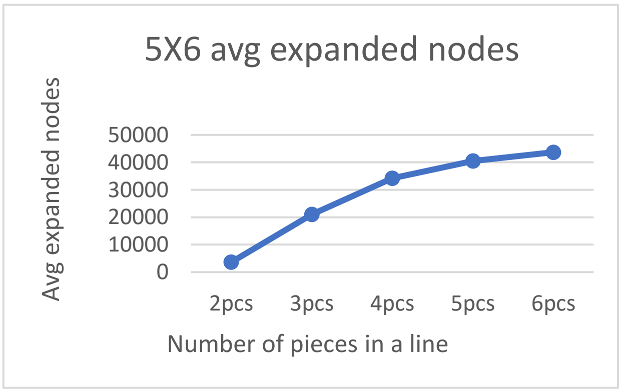

Fig3: Relationship of number of pieces vs avg expanded nodes.

2. EVALUATION OF MINIMAX WITH PRUNING

The evaluation of minimax with the alpha beta pruning implementation starts with comparing the **board size** with the number of **nodes expanded** and **pruned**. A depth of 5 is used for the search. The graph illustrates a **linear increase** in the expanded nodes per game until a board size of **6X6** and then increases **exponentially.** The average number of branches pruned sharply increases until a reasonable board size of **5X5** and then **sharply decreases** on the interval from the **5X5** and **6X6** board states.

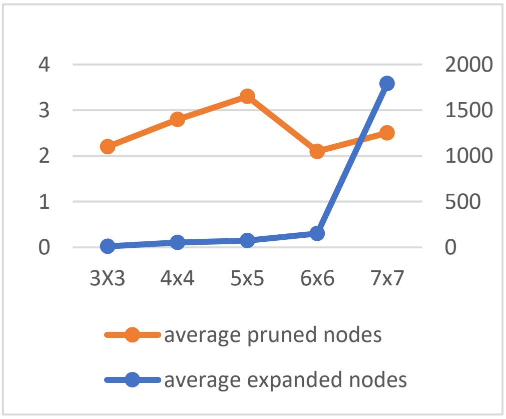

Fig 4: Relation between board size, number of expanded and pruned branches.

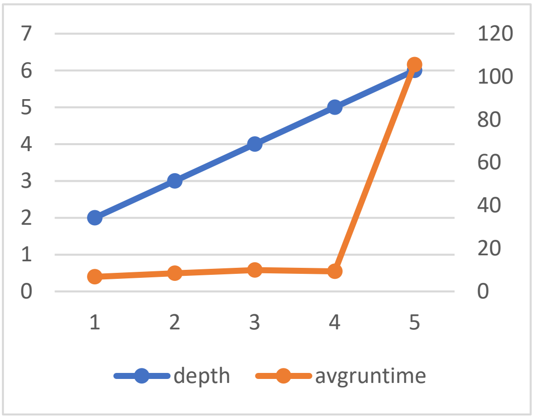

Fig 5: Relation between depth and average runtime

Other metrics such as board size and the average runtime are also considered during the evaluation of the algorithm. The graph shows a linear relationship between these two metrics. There is a similar relationship between the number of expanded and pruned nodes in the search.

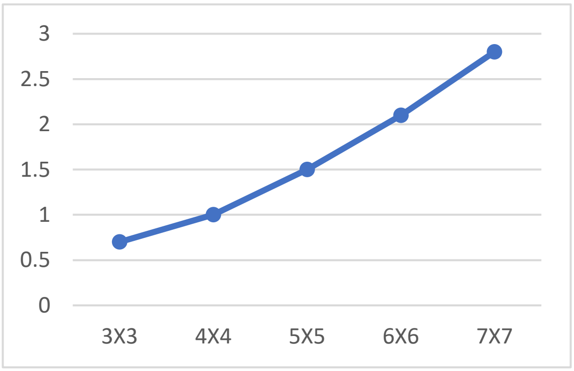

Fig 6: Relation between board size and average runtime

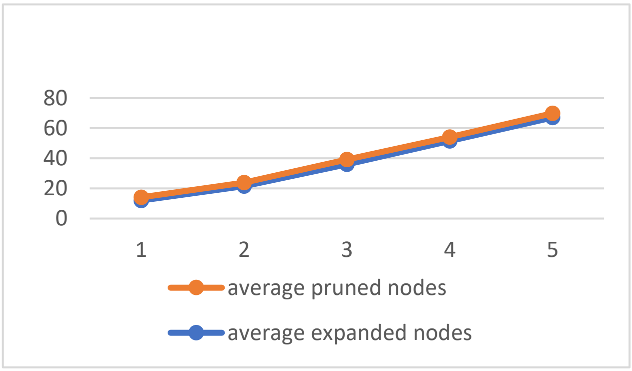

Fig 7: Relation between average number of expanded and pruned nodes

3. DISCUSSIONS AND CONCLUSION

The concepts of the normal minimax and the minimax with alpha beta pruning differ based on the number of nodes expanded and branches pruned during the search. The alpha beta pruning strategy significantly reduces the number of nodes expanded during the search thus decreasing the runtime significantly thus providing greater efficiency.

Additional features such as heuristics were also added, which improves the efficiency of the minimax algorithm by helping our **coursework player (Player X)** make optimal moves. This was a crucial technique which was used to improve the win rate of our player significantly close to **100 percent** thus making optimal decisions even on **high board states**. The **addHeuristics()** method is designed to block the opponent from winning in **most** situations. The score is incremented accordingly. The method is designed to check every possible move made by the player and strategically checks the required number of pieces required to win accordingly and assigns a score. However, since the function was designed to iterate through multiple moves, the time complexity therefore gets affected making it to increase.

If we compare the average number of expanded nodes with normal minimax and the one with the pruning logic, we see a slight variation in **Fig1** and **Fig6.** In **Fig1,** the average number of expanded nodes sharply increases after a certain board size **exponentially** to a high amount since it explores through the entire search tree. Whereas, for **Fig6,** there is always a **linear** increase in the number of the average nodes expanded thereby indicating that the average number of nodes expanded nodes reduce and increases at a **linear rate.**

Optimization techniques such as transposition tables were used which store the previously computed states of the board which significantly reduces the number of nodes expanded and pruned during the minimax with the alpha beta pruning strategy. They also help in reducing the runtime of the algorithm.

To conclude, the evaluation of the minimax algorithm in an efficient **Connect 4** game discusses the effectiveness of the minimax algorithm along with designing data structures such as a transposition table further contribute to reducing the computational load significantly.

REFERENCES

1. GeeksforGeeks (2022). Minimax Algorithm in Game Theory | Set4 (Alpha Beta Pruning). Retrieved from **https://www.geeksforgeeks.org/minimax-algorithm-in-game-theory-set-4-alpha-beta-pruning/**

2. Baeldung (2022) Minimax Algorithm
Retrieved from
https://www.baeldung.com/cs/minimax-algorithm

3. Professional AI (2022). Alpha-Beta-Pruning. Retrieved from
https://www.professional-ai.com/alpha-beta-pruning.html