**An Intelligent Chess Player using an Alpha-Beta Depth-First algorithm (designing & implementing at least 2 heuristic functions)**

|  |  |  |  |
| --- | --- | --- | --- |
| names | IDs | level | major |
| نورهان فريد عبد المقصود | 202001012 | 3 | IS |
| نسرين حسين محمد علي | 202000997 | 3 | IS |
| امنية امير فاروق عبد الحميد | 202000152 | 3 | IS |
| يارا مصطفي محمد عبد الغني | 202001051 | 3 | IS |
| سيف احمد عبد العليم | 20180277 | 3 | IS |
| يوسف وحيد عبد الحميد فرحات | 202001115 | 3 | IS |

Team ID: 57

Project idea and overview:

our project is to make a chess ai using the alpha beta depth first algorithm and Chess is a board game between two players. It is sometimes called Western chess or international chess to distinguish it from related games, such as xiangqi (Chinese chess) and shogi (Japanese chess).

Chess is an abstract strategy game and involves no hidden information. It is played on a chessboard with 64 squares arranged in an eight-by-eight grid. At the start, each player controls sixteen pieces: one king, one queen, two rooks, two bishops, two knights, and eight pawns. The player who moves first controls white pieces, and the other controls black pieces. The object of the game is to checkmate the opponent's king, whereby the king is under immediate attack (in "check") and there is no way for it to escape. There are also several ways a game can end in a draw as if the king has no more legal moves but is not checked or if there is no way any one could win with the left peices(usually a combination of three or two white king , black king and either a bishop or a knight for any).

**similar applications :**

there is many chess games on the internet but if to list them as for the best it will diffenitely include

1 Magnus trainer

2 learn chess with Dr wolf

3 chess.com

4 shredder chess

5 lichess

**A Literature Review of Academic publications:**

1-iCat, the chess player: the influence of embodiment in the enjoyment of a game:

This paper presents an experiment that evaluates and compares the user enjoyment when playing a game of chess in two situations: against a physically embodied robotic agent and against a virtually embodied agent, displayed on screen. The results of the study suggest that embodiment has implications on user enjoyment, as the experience against a robotic agent was classified as more enjoyable than against a virtually embodied agent.

2- Literature Survey of Chess Engines:

- Chess has come a long way since making its appearance on a computer and every year there is a new chess engine created that works better than the previously existing ones. But all these engines have one thing in common, they all work on some variant of the MinMax algorithm. The game tree is made shorter and more efficient and ways to play around with the graphical user interface are invented. The search tree is then pruned to save the time that the machine needs to compute the next best move. Thanks to all this, chess engines have come a long way and are now known all around the world to defeat some of the best Grandmasters of the world with their own strategies. But the AI in chess still has a long way to go before it can be declared as the fastest and the best chess playing machine.

https://www.ijert.org/research/literature-survey-of-chess-engines-IJERTCONV5IS01199.pdf

3- The Role of Chess in Artificial Intelligence Research:

Our eminent researchers including John McCarthy, Allen Newell, Claude Shannon, Herb Simon, Ken Thompson and Alan Turing put significant effort into computer chess research. Now that comput­ ers have reached the grandmaster level, and are beginning to vie for the World Championship, the AI community should pause to evaluate the sig­ nificance of chess in the evolving objectives of AI, evaluate the contributions made to date, and assess what can be expected in the future. Despite the general interest in chess amongst computer scien­ tists and the significant progress in the last twenty years, there seems to be a Jack of appreciation for the field in the AI community. On one hand this is the fruit of success (brute force works, why study anything else?), but also the result of a focus on performance above all else in the chess community. Also, chess has proved to be too challenging for many of the AI techniques that have been thrown at it. We wish to promote chess as the fundamen­ tal test bed recognized by our founding researchers and increase awareness of its contribution to date.

<https://www.researchgate.net/publication/220814778_The_Role_of_Chess_in_Artificial_Intelligence_Research>

4- ECM3401 Literature Review. A History of Computer Chess:

This paper provides a look into the rich history of computer chess, ranging from the mid-1940s to

present day. It gives a historical perspective of how computer chess has changed over the years, and

the ways in which it has remained the same.

The main body of the review is a comparison of the algorithms behind computer chess programs, a

description of how they work and the history of their creation. This is further split into three sections

which describe the main issues affecting the development of a computer chess program. These are

board representation, tree searching, and positional evaluation.

Firstly we look at board representation, and show how the most basic idea of a chess board can

be turned into a working board representation. We then look at more recent approaches, and how

these compare and contrast.

Next is the largest section of the report - the Tree Searching section. This gives a brief introduction

into how a computer will go about searching for the right move, and give an overview of the algorithms

that have been created to do this.

Lastly is the positional evaluation section, which details how a computer chess program values

different board positions.

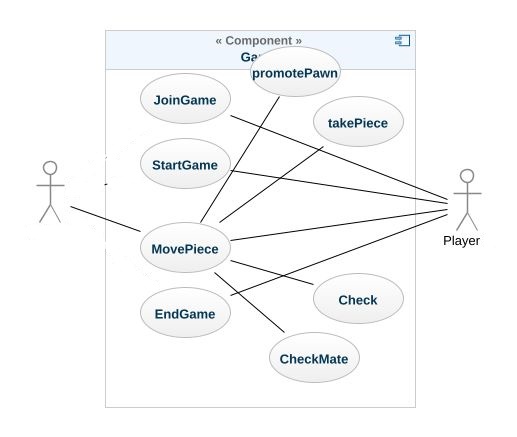
<https://silo.tips/download/ecm3401-literature-review-a-history-of-computer-chess>

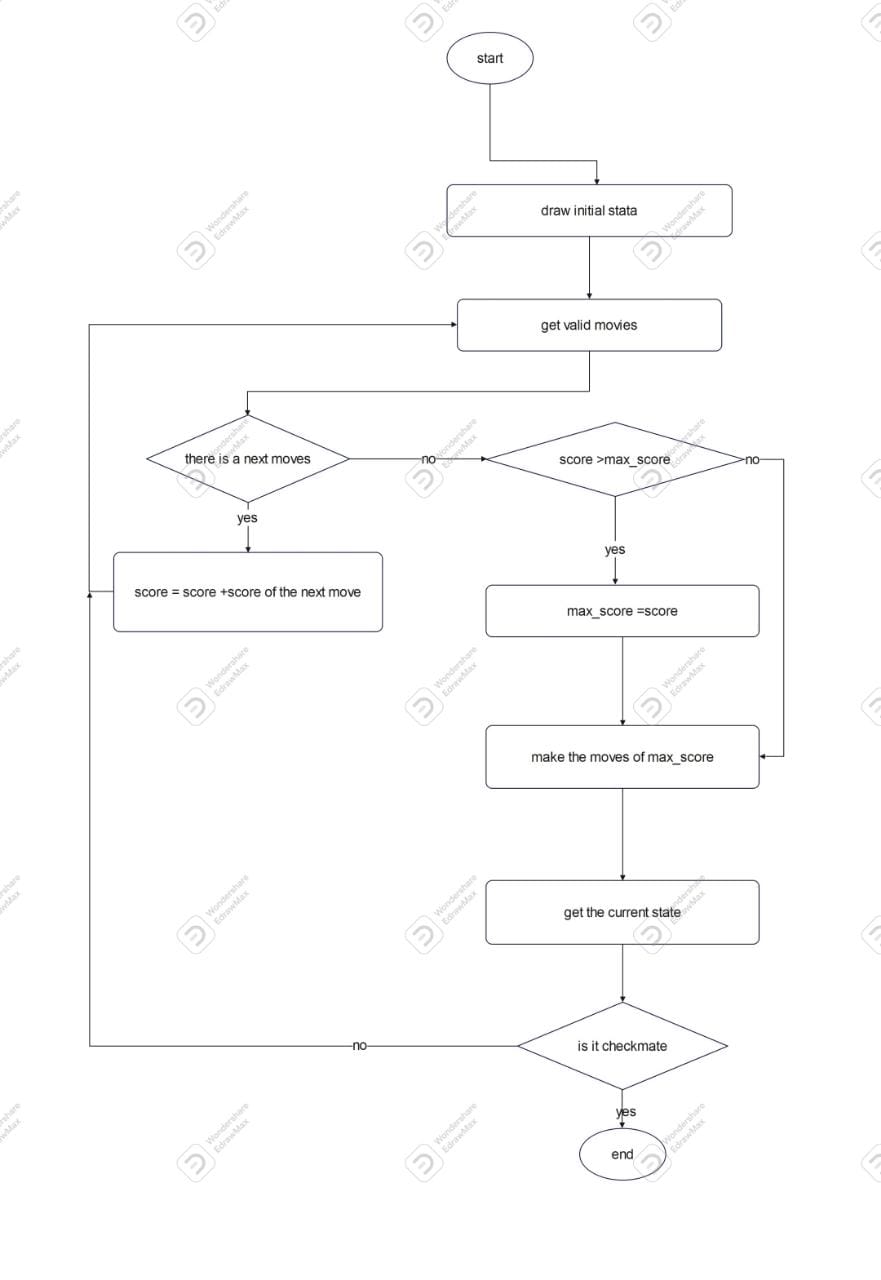
5- A chronology of computer chess and its literature:

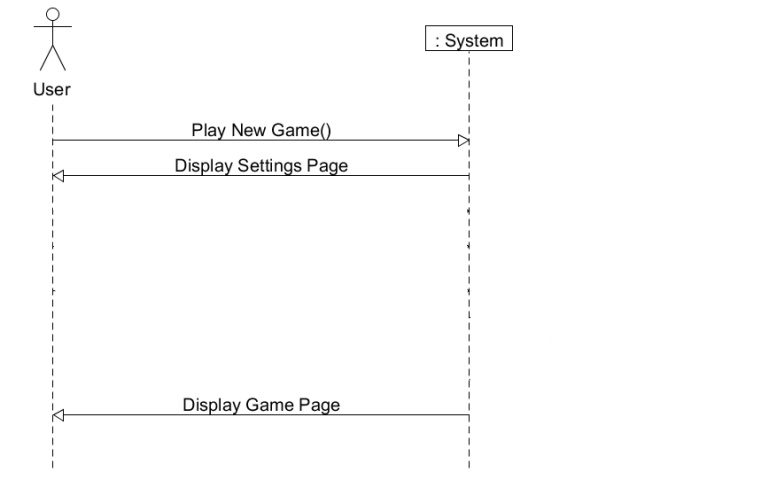
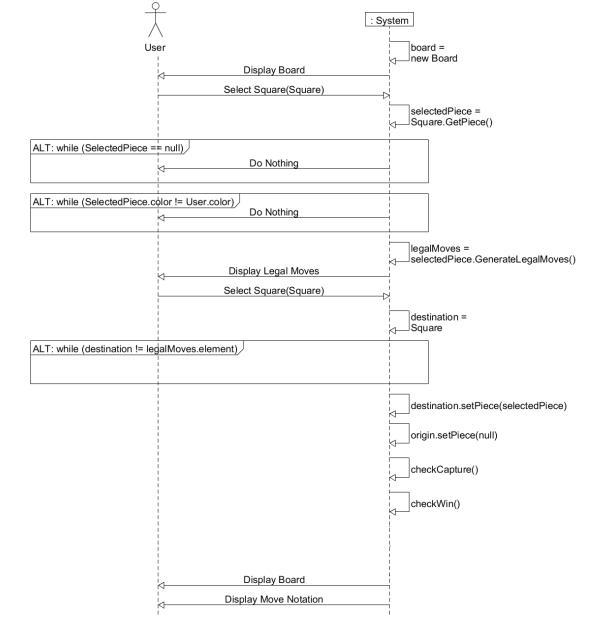
It can be seen that a great deal of worthwhile material has now been generated about computer chess. There is also quite a bit of nonsense by persons who have never built a program. Several groups with excellent programs have done little publishing, although I can hardly blame them since their work requires much time and is usually unsupported by any funding agencies. Certain staples have given rise to duplication: All but one of the books published explain the depth-first alpha-beta procedure.

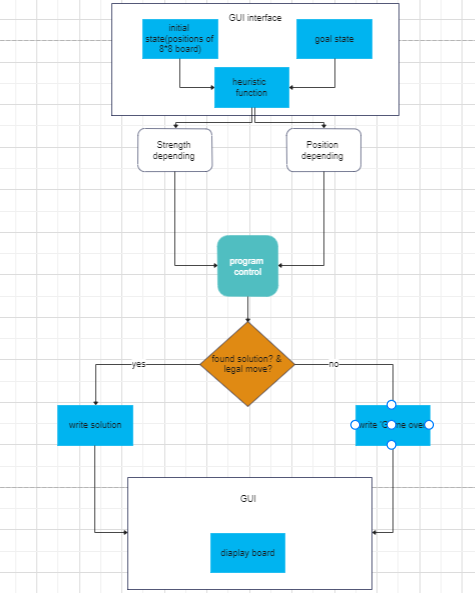
We expect that by far the largest portion of our readers will be scientists interested in updating their knowledge of the subject. To those few who are considering building their own program a word of caution: It appears that all the simple things have been done, and unless you merely want to re-invent the wheel, you must be prepared to spend at least 2 man-years of effort to reach the state-of-the-art with your own program. And to those few laymen that may see this: I hope I have delineated a bit more precisely than the media do, just what computers are capable of doing with chess circa 1978.

<https://www.sciencedirect.com/science/article/abs/pii/S0004370278800125>

**Proposed Solution & Dataset (Phase**







Experiments & Results

We used 2 heuristics in our project:

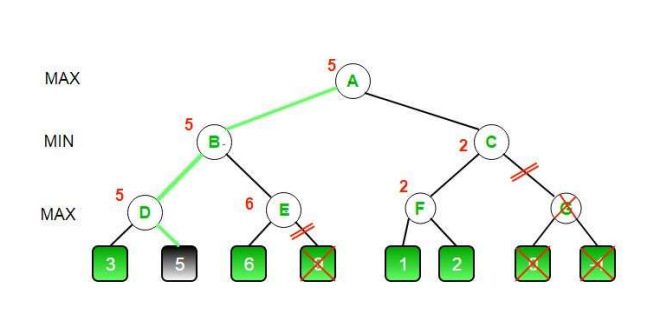
1. Strength depending heuristic
2. 2- Position depending heuristic

**Strength depending heuristic:**

With this one we assigned each piece with a score and depending on that score our Ai decides what piece to capture and what piece not to thus our Ai behavior tends to capture high value pieces and avoid exchanging a high value piece for one that has less score without good positioning and weird moves if there were no good captures ahead and no need to defend itself.

**Position depending heuristic:**

With this one we assigned each piece with a position array that defines which position is better for this specific piece for example: It’s better for the knight to stay in the middle of the board so we made his position score to the borders low so that the knight avoids being cornered to the border if not necessary. With that heuristic our Ai focused on being in better positions with no real interest in capturing until it’s necessary Our final scoring method was a combination between the two heuristics with the priority for the strength over the position and that gave us a better Ai that positions well and consider capturing and checking and defending. We also made our Ai faster by using alpha beta pruning and the table below shows the difference in the number of moves evaluated with and without alpha beta pruning with the same game state

Analysis, Discussion, and Future Work 1 Alpha-Beta pruning Is a search algorithm that seeks to decrease the number of nodes that are evaluated by the minimax algorithm in its search tree? It is an adversarial search algorithm used commonly for machine playing of twoplayer games (Chess in this case). It stops evaluating a move when at least one possibility has been found that proves the move to be worse than a previously examined move. It returns the same moves as minimax would, but prunes away branches that cannot possibly influence the final decision.

**Advantages and Disadvantages of ALPHA-BETA Prunin:**

**1-ADVANTAGES:**

• Allows elimination of the search tree branches.

• Limits the search time to more promising sub-trees, which enables a deeper search.

• Reduces computation and searching during minimax algorithm. Prevents the use of additional computational time, making the process more responsive and faster.

**2-DISADVANTAGES:**

• It does not solve all the problems associated with the original minimax algorithm.

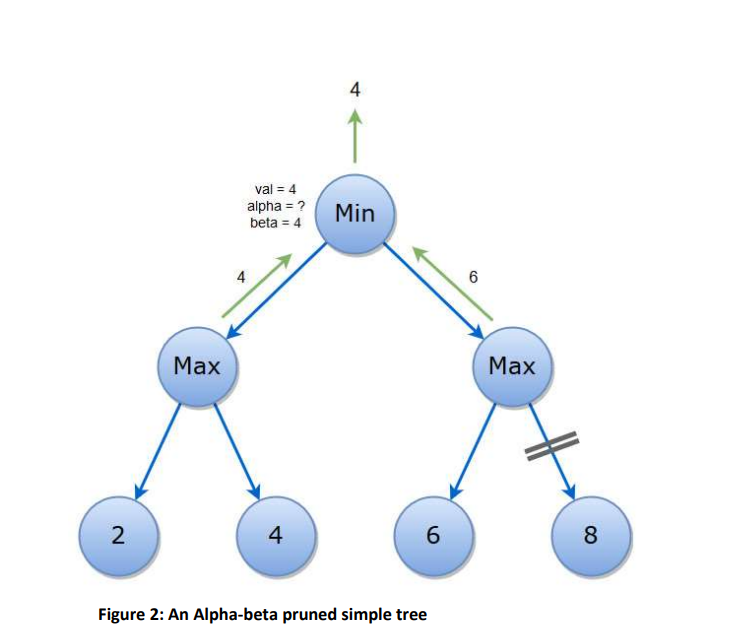
• Requires a set depth limit, as in most cases, it is not feasible to search the entire game tree (in case of chess exists total of 10^120 possible chess moves or nodes).

• Though designed to calculate the good moves, it also calculates the values of all the legal moves.

2. ALPHA-BETA

2.1 Tree Structure

We will use a search tree to look at the board and evaluate specific board states. Our search algorithm will look at this data structure and encode the potential nodes into the tree. Each connecting branch represents a legal move that can be made from one game state to another. Each node can have as many children, or branching nodes, as there are legal moves with, with the single root node represents the beginning of the game. The specific type of search tree in use is the adversarial search tree. We can use an adversarial tree because chess is a two-player, zerosum, perfect information game. Zero-sum means that any loss to one player is advantage to the other and perfect information means that nothing is hidden or left to chance. A search tree has two defining parts, the branching factor and the depth. The branching factor is the number of possible options that can be taken from a node. The depth is the number of moves that will be made in a specific line. For example, the game of chess has 20 possible opening moves, or child nodes. The next player also has 20 possible moves, making 20 the branching factor. Therefore, after each player has moved once there are 400 potential game states, or nodes. The result of this for the tree is a root node that has 20 child nodes which, in turn, each have 20 child nodes. This means that with a depth of 2 the branching factor of 20 at each stage will leave us with 400 possible nodes. The nodes in the tree are evaluated according to a linear scoring polynomial. This is what Alpha-Beta will use to find its choice of move



**Alpha-Beta:**

is an improvement over naive Minimax. It eliminates, or prunes, branches that are guaranteed to be worse than what has already been considered. We can use Figure 2 to see a simple example of Alpha-Beta pruning with the result... Evaluating the tree begins the same as Minimax, from left to right. This also uses that adversarial search tree with Min and Max players. Searching through the left tree will yield 4 as the Max value by default as the first node to be evaluated setting the alpha value to 4. Next we encounter 8 and, while it is better then 4, we know that the next player will choose the best move and so choose the 2 node making it irrelevant. Brining the 4 back up to the root node sets alpha to 4 as it is the best option so far. Going down to 6 as the first choice in the second branch sets this branches beta to 6 since it the lowest option it has seen so far. The program then can cut off all remaining analysis of the branch as it knows that the beta is less than the alpha or 6 < 8, making it unnecessary calculation time to look at the remaining nodes in the branch. This pruning is what makes Alpha-Beta an improvement over Minimax.

**Evaluating Position:**

Once the computer can represent the board the next stage is evaluating the position. Each node that is evaluated through Alpha-Beta is ranked. Remember though that each evaluation the algorithm computes is not the current state but upcoming states of the game. In the same vein as the representation the method used to store and read will greatly affect the time spent on analysis.

**future modifications:**

the AI can observe how the player play and determine his level and paly at the same level as the player

and we can also teach the AI from a data set of book moves to help it start the game with the best position

https://drive.google.com/drive/u/1/folders/1Vf8fUM8iR2\_FHGcOelrLZZwgx6Mxs8QK