**Documentation**

**Introduction and Overview**

**- Project idea and overview.**

Our project idea is An Intelligent Go Player using an Alpha-Beta Depth-First algorithm.

Overview about what is Go Player game ad it’s rules:

* **Legal actions**: at each turn, the player can only place the stone on one of opponent's liberties, unless the player has any self-group to save.
  + If there exists any opponent's group that has only one liberty, the legal actions will be these liberties to cause a direct win.
  + Else, if there exists any self-group that has only one liberty, the legal actions will be these liberties to try to save these groups. If sadly there are more than one of these liberties, the player will lose in the next round :(
  + Else, there are no endangered groups for both players; the player should place the stone on one of opponent's liberties.
  + Suicidal moves are not considered as legal actions.
* **Winning criteria** (one of the following):
  + You remove any opponent's group.
  + There are no legal actions for the opponent (this happens around 1.6% for random plays).

Overview about the alpha beta algorithm:

Alpha-Beta pruning is not actually a new algorithm, rather an optimization technique for minimax algorithm. It reduces the computation time by a huge factor. This allows us to search much faster and even go into deeper levels in the game tree. It cuts off branches in the game tree which need not be searched because there already exists a better move available. It is called Alpha-Beta pruning because it passes 2 extra parameters in the minimax function, namely alpha and beta.

**- Applications**

‘Many Faces of Go’ is a desktop application that developed to Play go against the computer andIt keeps track of your record and playing strength.

**- A Literature Review**

As due to the lack to find papers/books/articles that talk about Go plyer game using Alpha beta Pruning algorithm so we try to understand alpha beta algorithm for two players game approach on connect for then try to do what we understand on Go player rules one of the paper that help us is :

1-Alpha-Beta Pruning in Mini-Max Algorithm –An Optimized Approach for a Connect-4 Game for <https://www.academia.edu/>

so they in that paperhas been implemented the game of Connect-4

using two algorithms and comparison between them is

studied. First algorithm is mini-max algorithm and second

one is mini-max with alpha beta pruning which is an

optimized version of mini-max algorithm. The study

revealed that for the same level of difficulty the two

algorithms behave very differently in terms of number of

iterations performed and time taken with alpha beta pruning

taking much less time and performing very few iterations

than mini-max to generate the game state.

**Proposed Solution & Dataset**

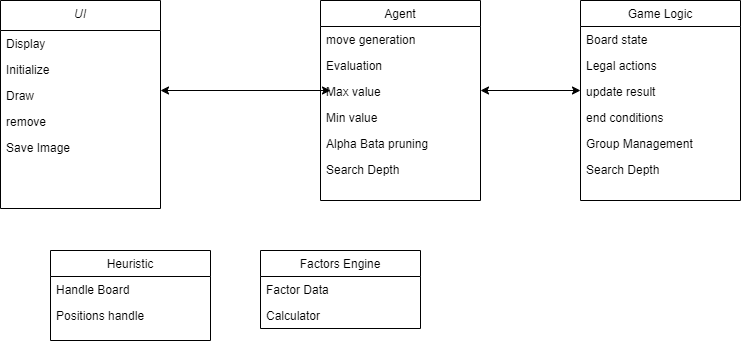
**- Main functionalities/features (from the users’ perspective) in your proposed software/solution (can be explained using a**

**use-case diagram).**

**Diagram

Description automatically generated**

Applied Algorithm



## We used an minimax algorithm and dipth first search algorithm n minimax algorithm and dipth first search algorithm Mini-Max Algorithm

* In AI, the Min-Max algorithm is mostly employed for game play. Chess, checkers, tic-tac-toe, go, and other two-player games are examples. This Algorithm calculates the current state's minimax choice.
* The game is played by two players, one named MAX and the other named MIN, in this algorithm.
* Both players FIGHT it, since the opponent player receives the smallest benefit while they receive the greatest profit.
* Both players in the game are adversaries, with MAX selecting the maximum value and MIN selecting the minimum value.
* For the exploration of the entire game tree, the minimax method uses a depth-first search strategy.
* For the exploration of the entire game tree, the minimax method uses a depth-first search strategy.
* The minimax algorithm descends all the way to the tree's terminal node, then recursively backtracks the tree.

## **Limitation of the minimax Algorithm:**

The biggest disadvantage of the minimax algorithm is that it becomes extremely slow while playing complex games like chess or go. This style of game contains a lot of branching, and the player has a lot of options to choose from. The minimax algorithm's drawback can be alleviated by using **alpha-beta pruning** , which we will explore in the next section. the depth to which the tree can grow.

Best way of learing.

# Alpha-Beta Pruning

* Alpha-beta pruning is a modified version of the minimax algorithm. It is an optimization technique for the minimax algorithm.
* As we have seen in the minimax search algorithm that the number of game states it has to examine are exponential in depth of the tree. Since we cannot eliminate the exponent, but we can cut it to half. Hence there is a technique by which without checking each node of the game tree we can compute the correct minimax decision, and this technique is called **pruning**. This involves two threshold parameter Alpha and beta for future expansion, so it is called **alpha-beta pruning**. It is also called as **Alpha-Beta Algorithm**.
* Alpha-beta pruning can be applied at any depth of a tree, and sometimes it not only prune the tree leaves but also entire sub-tree.
* The two-parameter can be defined as:
  1. **Alpha:** The best (highest-value) choice we have found so far at any point along the path of Maximizer. The initial value of alpha is **-∞**.
  2. **Beta:** The best (lowest-value) choice we have found so far at any point along the path of Minimizer. The initial value of beta is **+∞**.
* The Alpha-beta pruning to a standard minimax algorithm returns the same move as the standard algorithm does, but it removes all the nodes which are not really affecting the final decision but making algorithm slow. Hence by pruning these nodes, it makes the algorithm fast.

….....................................................................................................................................................

In our project we used two heuristic function

1.heuristic 1: here we call the max function

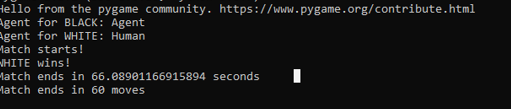
That the agent try to maxmize it’s winnig by

2.heuristic 2: and here we call the min function

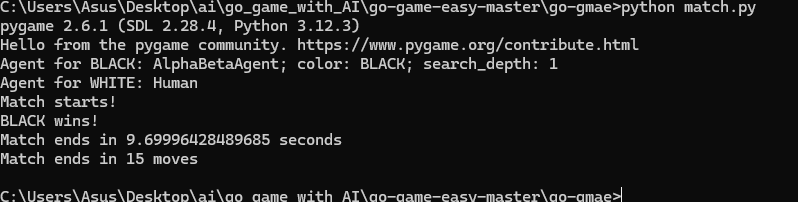
That the agent try to minimize it’s loss

Experiments & Result

In first experiment we use huristic function min\_value() that try to minimize its loss as shown in image:



And in second experiment we use huristic function max\_value() that try to beat human aggressively as shown in image:



so max\_value can finish match in 9.69 seconds with only 15 moves.

Analysis, Discussion, and Future

- Analysis of the results, what are the insights?

2 huristic function do good work and we can call The Agent is very smart

In Go Game

What are the advantages / disadvantages?

### Advantages:

1. **Agent-Human Comparison**:
   * This scenario shows a competition between an AI agent and a human player. It can help benchmark the AI's performance against human decisions.
   * Results, such as "WHITE wins," allow you to evaluate AI weaknesses or strengths.
2. **Performance Metrics**:
   * The time taken (66 seconds) and the number of moves (60 moves) provide clear metrics to analyze:
     + Efficiency of the gameplay.
     + Time complexity for AI decision-making.
3. **Testing AI Behavior**:
   * If this is a turn-based strategy game, the AI can be improved by analyzing its performance (e.g., how quickly it loses, decision-making weaknesses, or strategies it struggles against).
4. **Human Involvement**:
   * Having a human as a baseline for performance ensures realistic comparisons rather than pure agent-vs-agent scenarios.

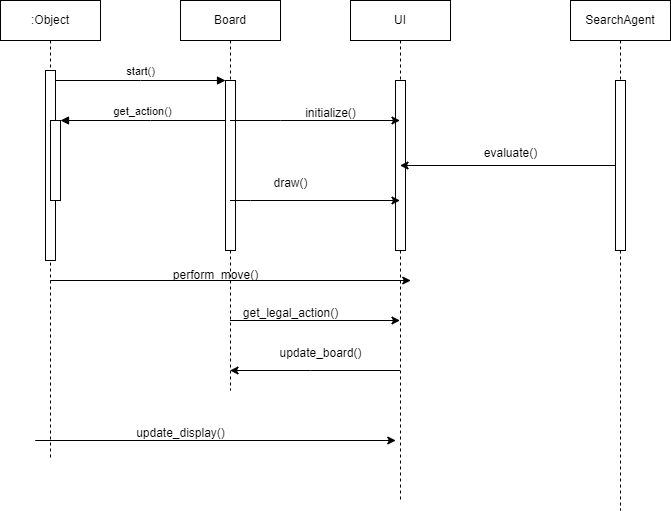
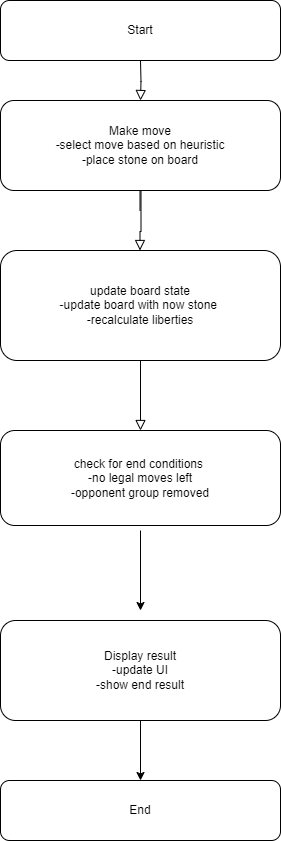
### Disadvantages:

1. **AI Limitations**:
   * If the AI loses consistently (e.g., here where "WHITE wins"), it might indicate weaknesses in the agent's strategy or training.
   * Time consumption (66 seconds) may also suggest inefficiencies in the agent's computation or decision-making algorithms.
2. **Human Variability**:
   * A human player's performance can vary due to fatigue, inconsistent strategies, or errors, making it difficult to draw absolute conclusions about AI performance.
3. **Scalability**:
   * This simple scenario may not be representative of more complex games, especially if the AI cannot scale to larger datasets or real-world conditions.
4. **Limited Data**:
   * Without more context (e.g., game type, AI algorithm used, human skill level), it is hard to generalize these findings.

- Why did the algorithm behave in such a way? What might be the future modifications you’d like to try when solving this problem?

-- Players Can use any Places be more free in choose place in game

-- find Another Huristic function that made Agent undefeatable

****

Githublink

<https://github.com/siecp-1001/go-gmae.git>

|  |  |
| --- | --- |
| name | id |
| بيشوي جورجي عطيه حنا | 202000216 |
| محمد سيد عنتر احمد | 20210786 |
| يوساب رمزي نصحي | 202001067 |
| عبدالرحمن شريف محمود عبدالعظيم | 202000516 |
| Ahmed Ayman Yahia sayed | 202000018 |
| مصطفي محمد فوزي طه | 202000914 |
|  |  |
|  |  |
|  |  |