**FAI project proposal**

**Team members: Alexander Stults, Purvik Patel, Vikram Bharadwaj, Mansi Pawar**

For our final project, we are trying to develop an AI agent to play [Stratego](https://en.wikipedia.org/wiki/Stratego). It is a strategy board game played between two players. Each player has 40 pieces with different ranks. The goal of the game is to capture the opposite player’s flag or capture other pieces till the opposition cannot make any further moves. There are 10^115 legal game states.

1.     **Initial Board Layout**

The first thing a player needs to do whenever a new game of stratego begins is to arrange his 40 pieces on his side of the board. Unlike traditional games like chess, stratego does not have any fixed starting positions for all the individual pieces. Each player must start with their setup and these starting setups define the fundamental strategy a player is going within that game. Coming up with good starting board positions is important for winning the game.

We will use the genetic algorithm and steepest hill algorithm to come up with a good starting board position for the AI agent. Starting position does not win us the game but helps us start the game strongly. There can be many good starting positions so there won't be a proper goal state, but we will try to implement it to start with a random state and improve it in a fixed amount of iterations using GA or steepest hill and start the game with that state. To determine if one state is better than the other, we will have to check if our flag is protected well, all the other pieces are arranged in a way where we have a good defense against the enemy team and we also have good attacking pieces in our front row to initiate our offense. Based on the setup reached by the algorithm, we will also figure out what strategy we should use to start the game, aggressive or defensive?

2.     **Movement/Attacking**

To come up with a strategy, we'll use search algorithms. As a result, they will search for the possibilities and choose the best move. We'll use the **A\* algorithm**, as well as **Breadth First Search** (BFS) and **Depth First Search** (DFS). The goal of these algorithms will be to determine the best set of moves to get them to the final condition. These algorithms will look for a set of moves based on those conditions.

For DFS, uses the minimum time when the target is nearer, i.e., stops the searching when the goal node reaches. In the worst cases also, there is a chance of reaching the target using many paths to find the goal state. The main reason to use the Depth-First Search is because of its space requirement in a linear manner while searching. For BFS, the shorter number of steps is required to reach the goal node. It guarantees the path the reach the goal. But it isn’t that optimal as it gets struck in the loop by using DFS. A\* finds the best path in minimum time and reduces time complexity.

If we are stuck at any step, we go reverse and start **backtracking**. It used the DFS method.

3.     **Inference of Enemy at Given Position**

One of the main strategic decisions within Stratego is whether to attack or not. This decision is keyed off of two main important factors: first, whether or not you are sure that you are stronger than the opponent, and second, whether the information to be gained is worth the risk of facing an unknown enemy piece. We will be using Bayesian inference/probability to attempt to determine the given identity of an unknown enemy piece. We should be able to utilize different observed behaviors and movements of the enemy to establish a probabilistic level of certainty across a range of possible enemy units. For instance, if an enemy at a given position has never moved, and the game is sufficiently advance, it is highly probable that the unit is either a bomb or the flag. These sorts of deductions and more will be explored by our implementation.

4.     **Strategy**

• Utility score based on a heuristic:

For every move that is made by either player, we will be calculating a heuristic score for that move to determine how effective the move is and will improve the chances of winning for the player. In this approach, each heuristic is assigned a utility score that can either increase as a rewarding mechanism after an improving move or decrease as a punishment mechanism after a worsening move. The utility score is updated after each step of the algorithm, and the heuristic with the best score can be selected as the default strategy.

• Minmax with alpha-beta pruning:

Every move of a player will lead to a game state which will be advantageous to one and disadvantageous for another player. The first player will try to maximize the heuristic score and is called the MAX player, while the second player, the MIN player, tries to minimize the heuristic score. The algorithm will recursively alternate between the MAX player and the MIN player until it reaches a terminal node. The game tree will be searched in a depth-first way meaning that the algorithm will search the deepest unvisited branch until a terminal node is reached. The value of the terminal node is then returned to the parent node. When all the branches of a node have been visited, the player can now choose the best branch from that node and return the value of that branch to the parent node again up to the root node.

5.     **Endgame / Scenarios**

* Small set of problems with discoverable 'obvious' actions.
* i.e. Red has only the flag left and it should be attacked.