AI Programming Project Report

Introduction:

To implement a Quarto agent we used Alpha-beta pruning and then we implement several evaluation functions. However, we used one of the evaluation function in our final version of the code.

A player in quarto game receives a piece from the server and then chooses a place on the board to put that piece and then chooses a piece for the opponent to play with. So there are two stages in every move and we define these two stages as a single move in out alpha beta. For example, for the first move on an empty board the branching factor will be 25*32 = 800.

Alpha-beta pruning:

By adding alpha beta pruning to the Minimax we can prune a large part of the tree. Most of the time this allows us to expand the tree one more layer than a simple Minimax in the same amount of time and space. In *ABP1* method we set the *depth limit* and then call the *ABPmax1* to start the alpha-beta. If our agent is starting the game, going through an alpha beta is useless instead we can choose a random piece.

To reduce space complexity, we used a simple trick in our implementation. We are just using a single board all the time because there is no need to create a new board for the next stage. We just can *undo* changes we made through recursions.

Evaluation Functions:

This is the main part of the problem. For finding a perfect evaluation function, we need experience and knowledge about the game and unfortunately we don't have enough time to do research about it. However, in this case we implemented three kinds of evaluation functions.

- **Eval1**: counting similar attitudes in each row and multiple them with some weights. For example, imagine a row with 4 pieces. We find the number of the common attitudes between them and then multiple this number by a weight. The result will is the value of this row. We are doing this repeat it for all rows, columns and diagonals and to calculate the final value of the board we sum all of these values. It is decided to use a weight in this evaluation function because a common attitude in a row of 4 pieces is more promising than 3.
- **Eval2**: The second function not only evaluates the board, but also considers the *not-played pieces*. This function finds which attitudes are common in 4 pieces line (4 pieces line = a row/columns/diagonal with a single empty place), then iterates through the remaining pieces and counts the pieces which can complete that line with. To calculate the value of the board we simply sum all the completing pieces

for each row. If there is a 4 pieces line but no not-played piece can complete it, we won't increase the value of the board. This evaluation seems to be more promising than the first one.

- **Eval3**: A simple Monto Carlo! Herein this evaluation function, we run Monto Carlo for each board 500 times. This number can change depending on the time limit.

Dynamic depth limit:

We used a simple way to have a dynamic depth limit. In the first rounds of the game we are trying to go down 2 or 3 levels, but as we get into the game we can go down for more than 7 or 8 layers. The number of the depth limit is depended on three factors:

1. Number of empty places on the board.
2. Number of remaining pieces.
3. Time limit.

To find the depth limit, we increase the *depthLimit* (default value is 2) by 1 until the following equation is true:

$$(e.r)^d < \propto .t$$

Where

e = Number of empty places on the board
r = Number of remaining pieces
d = Depth limit
α = Constant variable to convert *Time* to *Operation number*
t = Time limit

Conclusion:

We preferred the second evaluation function over the others. The second one usually win against the other two, however, we are not sure about it because we don't have an appropriate test bench. We only can compare these three versions together; so, we are not sure which one will be the best in general.

We noticed that even if both players don't play optimal, most of the games will be draw because the original quarto game has been extended to 5x5 (32 pieces) and for last moves, there are still a lot of unplaced pieces. While in the original game (16 pieces) we don't have many options for last moves.