Bot Documentation

Jordan Aherne – 17482772

Rafal Koziel – 17472032

Oisin McPhillips – 17409914

The method we used to calculate the best move for our bot is we assigned various weights to moves depending on features of the move which we thought were valuable when playing a game of backgammon. Following are the methods which we used to achieve this and the reasons why we implemented them.

findMove()

Our find move method takes all possible plays and passes each play into another function. This function checks what our evaluate play method returns to see which move had to highest weight and then stores that value.

evaluatePlay()

Evaluate play takes in a play and goes through each move in play to check some features such as is the move a hit or can you move off the board on that move. For the other features we make a possible move in a temporary board and then pass this board to different methods to check other features of the total play.

calculateStrongPosition()

This method checks if a point has been made on a pip where a point is when two or more checkers are on a pip. If a point has been made we add a number to our total weight and if a point has been made on particular pips that we thought meant a stronger position we added a higher weight.

calculateBlot()

This method checks if a blot has been made in this play and if so take away a number from the total weight, showing that this move is worse. If a blot has been made between certain pips that we thought were bad pips to make a blot on we took away a bigger number from the total weight showing that this move is very bad.

calculateStrong()

This method checks if points have been made in the home board and if so add a number to the total weight. The move points made the higher the weight that is added.

calculateHit()

This method checks if we have points on our home board and if we have more that four add a high number to the total weight, if not add a number for normal hit.

getOdds()

This method is to check the odds of you winning the game so that you can make a decision to make, accept or decline a double. The way we calculated the odds for our chance of

winning was to take the total distance of all our checks to the end of the board and the total distance of your opponents checkers to the end of the board. We then do our distance over opponents distance and multiply it by one hundred to get an estimate of our chances of winning.

checkOdds()

We use the odds from getOdds to make a decision on whether to double or not on each move. This method just checks if the odds our in our favour and if so then propose a double.

getDoubleDecision()

We use odds from getOdds again to check if we should accept or decline a double.

To determine the weights for the features of each move we just picked values we thought fit and then tested our bot against itself while adjusting weights of one of the bots to try improve it, if we saw an improvement we would change the other bots weights to match and then try adjust one bot to improve from there and so on until we were happy with the methods and weights we had.