The Bidding Game

By,

Satish Kumar (1MS13CS095)

Swapnil Jain (1MS13CS121)

Himanshu Jaju (1MS13CS139)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

M S RAMAIAH INSTITUTE OF TECHNOLOGY

Jan-May 2016

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

M S RAMAIAH INSTITUTE OF TECHNOLOGY

Bengaluru – 54

CERTIFICATE

This is to certify that ***Satish Kumar (1MS13CS095), Swapnil Jain (1MS13CS121), Himanshu Jaju (1MS13CS139)*** *have completed the* ***"The Bidding Game"*** *in partial complement for the course on* ***Artificial Intelligence.***

We declare that the entire content embodied in this B.E. 6th Semester Project report contents are not copied.

Guided by:

Dr. Jagadish S Kallimani

(Assistant Professor, Dept. of CSE)

ACKNOWLEGEMENT

I express my sincere gratitude to Dr. Jagadish S Kallimani, Dept. of Computer Science and Engineering, MSRIT, for his mentorship and guidance, continuous encouragement, supervision and constant reminders throughout the course of present work.

Signature(s) of Students

Satish Kumar (1MS13CS095)

Swapnil Jain (1MS13CS121)

Himanshu Jaju (1MS13CS139)

INDEX

ABSTRACT

The Bidding Game is a combinatorial game in which, rather than alternating moves, the two players bid for the privilege of making the next move. We find optimal strategies for both the case where a player knows how much money his or her opponent has and the case where the player does not.

Two players are at the opposite end of the straight line as shown in the picture above. An expensive scotch (S) is kept in the middle at position #5. The players start the game with 100 dollars in hand.

The first player makes a secret bid followed by a secret bid by the second player. The bottle moves one position closer to the winning bidder. In the event of the same bid, the bottle moves closer to the player who has the draw advantage. Draw advantage initially starts with the first player, and it alternates every time a draw is encountered i.e , The first draw is won by the first player. The second draw if it occurs is won by the second player and so on. The winning bid is deducted from the player's hand, the loser keeps his bid. Each bid must be greater than 0 dollar. In the case when there's no money left, the player has no choice but to bid 0 dollar. Only integral bids are allowed.

The result of the game is determined by the cumulative results of all the different rounds both the players play with each other. In each round the bottle either moves left , right or stay at his location , finally the player who gets the bottle win the game , if no one gets it, the game ends in draw.

1.0 INTRODUCTION

There are two game theories. The first is now sometimes referred to as matrix game theory and is the subject of the famous von Neumann and Morgenstern treatise [1944]. In matrix games, two players make simultaneous moves and a payment is made from one player to the other depending on the chosen moves. Optimal strategies often involve randomness and concealment of information.

The other game theory is the combinatorial theory of Winning Ways [Berle 1982], with origins back in the work of Sprague [1936] and Grundy [1939] and largely expanded upon by Conway [1976]. In combinatorial games, two players move alternately. We may assume that each move consists of sliding a bottle from one vertex to another along an arc in a un-directed graph as shown Fig 1.1. A player who cannot move loses. There is no hidden information and there exist deterministic optimal strategies.

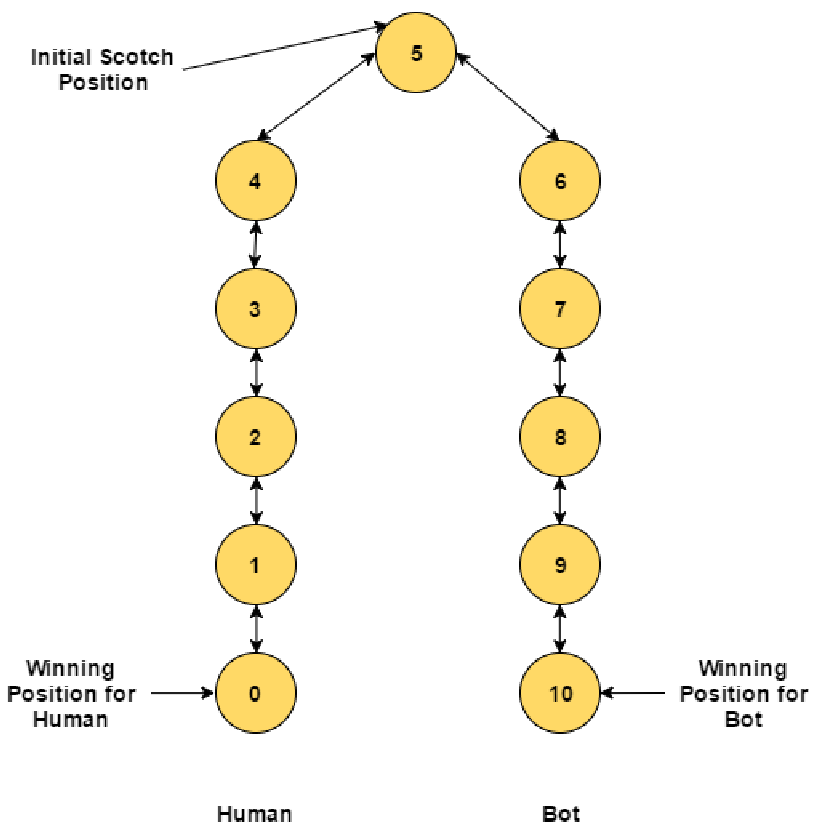

Fig 1.1

In this bidding game which we tried to develop is a games that share some aspects of both sorts of game theory. Here is the set-up: The game is played by two players (Player1 and Player2), each of whom has some money. There is an underlying combinatorial game in which a token rests on a vertex of some un-directed graph. There are two special vertices, denoted by 0 and 10; Player1's goal is to bring the token to 0 and Player2's goal is to bring the token to 10. The two players repeatedly bid for the right to make the next move. One way to execute this bidding process is for each player to write secretly on a card a nonnegative real number no larger than the number of dollars he or she has; the two cards are then revealed simultaneously. Whoever bids higher pays the amount of the bid to the opponent and moves the token from the vertex it currently occupies along an arc of the un-directed graph to a successor vertex. Should the two bids be equal, the tie is broken by a toss of a coin. The game ends when one player moves the token to one of the distinguished vertices. The sole objective of each player is to make the token reach the appropriate vertex: at the game's end, money loses all value. The game is a draw if neither distinguished vertex is ever reached.

There is never a reason for a negative bid: since all successor vertices are available to both players, it cannot be preferable to have the opponent move next. That is to say, there is no reason to part with money for the chance that your opponent will carry out through negligence a move that you yourself could perform through astuteness.

A winning strategy is a policy for bidding and moving that guarantees a player the victory, given initial data where the initial data include where the bottle is, how much money the player has, and possibly how much money the player's opponent has for bidding in the next round.

2.0 LITERATURE SURVEY

Given a starting vertex v, there exists a critical ratio R(v) such that Player1's has a winning strategy if Player1's share of the money, expressed as a fraction of the total money supply, is greater than R(v), and Player2 has a winning strategy if Player1's share of the money is less than Palyer2(v). According to Richman [1] this is not so surprising in the case of acyclic games, but for games in general, one might have supposed it possible that, for a whole range of initial conditions, play might go on forever.

There exists a strategy such that if a player has more than R(v) and applies the strategy, the player will win with probability 1, without needing to know how much money the opponent has.

In proving these assertions, it will emerge that a critical bid for Player1 is R(v) − R(u) times the total money supply, where v is the current vertex and u is a successor of v for which R(u) is as small as possible. A player who cannot bid this amount has already lost, in the sense that there is no winning strategy for that player. On the other hand, a player who has a winning strategy of any kind and bids R(v) − R(u) will still have a winning strategy one move later, regardless of who wins the bid, as long as the player is careful to move to u if he or she does win the bid.

It follows that we may think of R(v)−R(u) as the fair price that Player1 should be willing to pay for the privilege of trading the position v for the position u.

Thus Richman define [1] (1 − R(v)) as the Richman value of the position v, so that the fair price of a move exactly equals the difference in values of the two positions. However, it is more convenient to work with R(v) than with 1 − R(v). We call R(v) the Richman cost of the position v.

For all the vertex v other than the distinguished vertices 0 and 10, R(v) is the average of R(u) and R(w), where u and w are successors of v in the digraph that minimize and maximize R(), respectively. In the case where the digraph underlying the game is acyclic, this averaging-property makes it easy to compute the Richman costs of all the positions, beginning with the positions 0 and 10 and working backwards. If the digraph contains cycles it is not so easy to work out precise Richman costs because the work Richman did is work only for directed graph.

Features included by the Richman

2.1 Richman Cost Function

The story of the bidding game is proposed by Richman who worked on such games, according to his work Richman defined a cost function[1] which denote a directed graph (V ; E) vertices and the edges as shown in figure 1.1 with a distinguished blue vertex b and a distinguished red vertex r such that from every vertex there is a path to at least one of the distinguished vertices. For all v belongs to V, let S(v) denote the set of successors of v in the digraph D, that is, S(v) = w belongs to V : (v; w) belongs to E. Given any function f : V → [0;1], he defined

$$f^+(v) = \max_{w \in S(v)} f(w) \quad \text{and} \quad f^-(v) = \min_{u \in S(v)} f(u).$$

The key to playing the Richman game [1] on D is to attribute costs to the vertices of D such that the cost of every vertex except r and b is the average of the lowest and highest costs of its successors. Thus, a function R: V → [0; 1] is called a Richman cost function if R(b)= 0, R(r) = 1, and for every other v belongs to V we have R(v) = 0.5(R^+(v) + R^-(v)). Richman costs is curious sort of variant on harmonic functions on Markov chains [Woess 1994], where instead of averaging over all the successor-values, we average only over the two extreme values. The relations R^+(v) >= R(v) >= R^-(v) and R^+(v) + R^-(v) = 2R(v) are used by the Richman in cost function. On the basis of Richman cost function we can say that is there is a winning strategy to win the game on the basis of the cost function which he did find out for each game played.

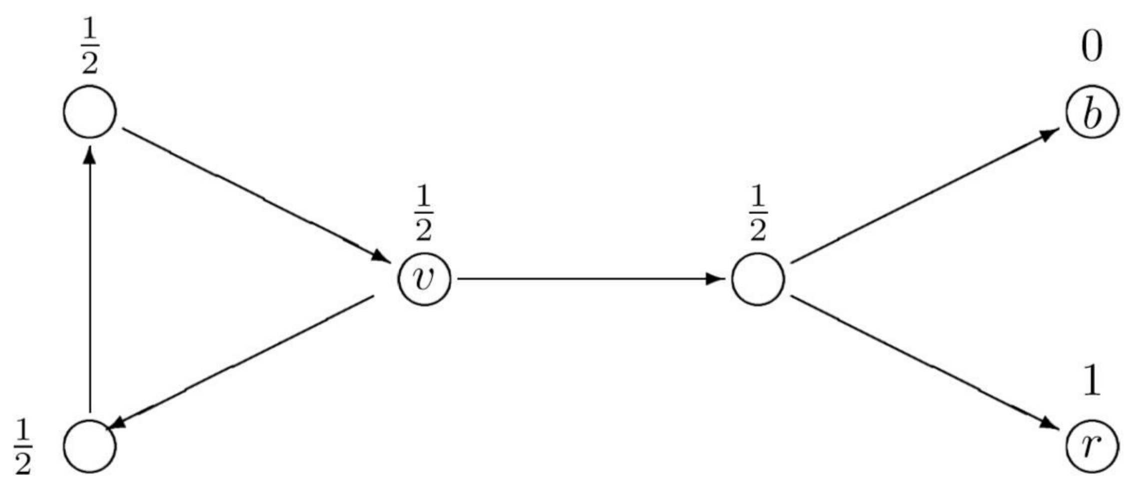

Fig 2.1 The digraph D and its Richman costs.

2.2 Continues Bidding

Continues bidding includes both the players player1 and player2 are made to play the game continuously, the winner at every step is determined by the comparison of the game threshold value T(G) with the value of b and the graph G is finite directed graph , where b is Player1 resources/(Player1 resources + Player2 resources). On the basis of the comparison there are three cases to determine the winner of every continues round.

Case 1: if b>T(G) , then the winner is Player1.

Case 2: if b<T(G) , then the winner is Player2.

Case 3: if b=T(G) , then the outcome depends on the flip coin.

Threshold value T(G) is calculated on the basis of the position of the player if Player1 can win from a position v then the value of T(G at v) is equal to 0. If the position v from where a Player1 cannot win then the value of the T(G at v) is equal to 1. The computation of T(G) is done using the Richman Cost Function[1].

2.3 Flip coin to break ties

Suppose the right to move the bottle is decided on each turn by the toss of a fair coin. The probability that Player1 can win from the position v in at most t moves is equal to Player1(v; t). Taking t to infinity, we see that Player1(v) is equal to the probability that Player1 can force a win against optimal play by Player2. That is to say, if both players play optimally, Player1(v) is the chance that Player1 will win. The uniqueness of the Richman cost function[1] tells us that Player1(v) has the chance to win. The probability of a draw is therefore zero.

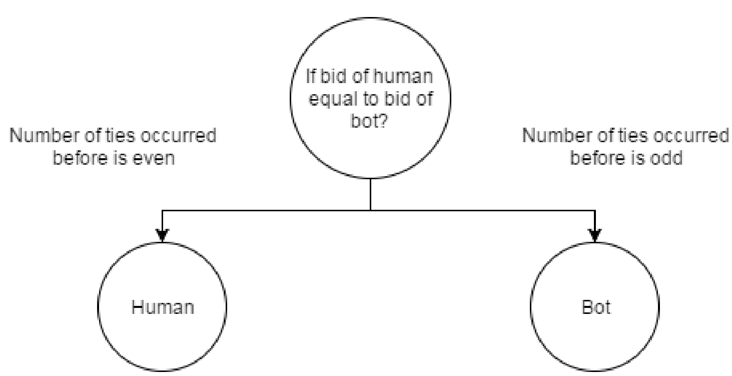

Fig 2.2 Tie State

If we further stipulate that the moves themselves must be random, in the sense that the player whose turn it is to move must choose uniformly at random from the finitely many legal options, we do not really have a game-like situation anymore; rather, we are performing a random walk on a directed graph with two absorbing vertices, and we are trying to determine the probabilities of absorption at these two vertices. In this case, the relevant probability function is just the harmonic function on the digraph D.

Another interpretation of the Richman cost, brought to our attention by Noam Elkies, comes from a problem about makings bets. Suppose you wish to bet (at even odds) that a certain Cricket team will win the World Series, but that your bookie only lets you make even-odds bets on the outcomes of individual games. Here we assume that the winner of a World Series is the first of two teams to win four games. To analyze this problem, Noam create a directed graph whose vertices correspond to the different possible combinations of cumulative scores in a World Series, with two special terminal vertices (team1 and team2) corresponding to victory for the two respective teams. Assume that your initial amount of money is Rs100, and that you want to end up with either Rs0 or Rs200, according to whether the team2 or the team1 wins the Series. Then it is easy to see that the Richman cost at a vertex tells exactly how much money you want to have left if the corresponding state of affairs transpires, and that the amount you should bet on any particular game is Rs200 times the common value of $R(v) - R(u)$ and $R(w) - R(v)$, where v is the current position, u is the successor position in which team2 wins the next game, and v is the successor position in which team1 wins the next game.

2.4 Incomplete knowledge

Surprisingly, it is often possible to implement a winning strategy without knowing how much money one's opponent has. Define Player1's safety ratio at v as the fraction of the total money that he has in his possession, divided by T(v) the fraction that he needs in order to win. Note that Player1 will not know the value of his safety ratio, since we are assuming that he has no idea how much money Player2 has this leads to incomplete knowledge about the game.

SupposePlayer1 has a safety ratio strictly greater than 1. Then Player1 has a strategy that wins with probability 1 and does not require knowledge of Player2's money supply. If, moreover, the digraph D is acyclic, his strategy wins regardless of tiebreaks; that is, with probability 1 can be replaced by definitely.

Theorem proposed by the Richman

Theorem 1. The digraph D has a Richman cost function f(v).

He introduce a function f(v; t), where he introduce a game-theoretic significance in which he assumes the initial values of f(b; t) = 0 and f(r; t) = 1 for all t belongs to N.

For v not belongs {b,r}; define f(v; 0) = 1 and f(v; t) = 0.5(R+(v; t − 1) + R−(v; t − 1)) for t > 0. It is easy to see that R(v; 1) <= R(v; 0) for all v, and a simple induction shows that R(v; t+1) <= R(v; t) for all v and all t >= 0. Therefore R(v; t) is weakly decreasing and bounded below by zero as t is not equal to infinity, hence convergent. It is also evident that the function v →lim f(v; t) where t tends to infinity , satisfies the definition of a Richman cost function.

Other way to come up to same conclusion by the use of Identify functions f : V (D) → [0; 1] with points in the |V (D)|-dimensional cube Q = [0; 1]|V (D)|. Given f belongs to Q, define g belongs to Q by g(b) = 0, g(r) = 1, and, for every other v belongs to V , g(v) = 0.5(f +(v) + f −(v)). The map f →g is clearly a continuous map from Q into Q, and so by the Brouwer fixed point theorem it has a fixed point. This fixed point is a Richman cost function. This Richman cost function does indeed govern the winning strategy.

Theorem 2. Suppose Player1 and Player2 play the Richman game on the digraph D with the token initially located at vertex v. If Player1's share of the total money exceeds f(v) = limt t→infinity f(v; t), Player1 has a winning strategy. Indeed, if his share of the money exceeds f(v; t), his victory requires at most t moves .Without loss of generality, money may be scaled so that the total supply is one dollar. Whenever Blue has over f(v) dollars, he must have over f(v; t) dollars for some. At t = 0, Player1 has over f(v; 0) dollars only if v = b, in which case he has already won.

Theorem 3. The Richman cost function of the digraph D is unique. It means that an edge (v; u) is said to be an edge of steepest descent if f(u) = $\bar{f}$(v).

The Bidding Game is already implemented by other bots around the world[2].

From the above Survey/Research which was done by Richman we inferred that

1. Richman work only for the directed graph.
2. Richman cost function tells whether there is a possibility of winning a game from the current position if there then you will always win the game.

3.0 IMPLEMENTATION DETAILS

Our aim in this bidding game is to find an optimal bid at every round of the game, so that the chances of our winning the game is high. To find this optimal bid, we need to pre-process the current state of the game and find the lowest bid which yields the highest probability of our victory.

The entire process of bidding an optimal amount can be divided into the following steps:

1. Pre-processing
2. Calculating Probability for each bid amount
3. Finding the optimal bid

3.1 Pre-processing

Pre-processing the current state of the game includes calculating:

a. Amount of money left with human
b. Amount of money with bot
c. Current position of the scotch bottle
d. Winner in case of a tie

$$Current\ State = \langle HMoney,\ BMoney,\ Pos,\ Tie \rangle$$

In the pre-processing step, we obtain information about the current state of the game using the given previous moves in the game. By simulating all the previous moves from the initial state, we can obtain all information on the current state of the game. Once the entire pre-processing is done, we move on to calculating the probability for every possible bid by the bot.

3.2 Calculating Probability for each bid amount

We assume that every bid is equi-probable for both the human and the bot. Using this assumption and the current state of the game, we can use dynamic programming to calculate the conditional probability of winning the game if we place an arbitrary bid x.

Let DP(HMoney,Bmoney,Pos,Tie) denote the probability of winning when the state of the game is such that :

- Human has HMoney amount of money left
- Bot has BMoney amount of money left
- Pos is the current position of the scotch bottle
- Tie denotes the winner in case of a tie

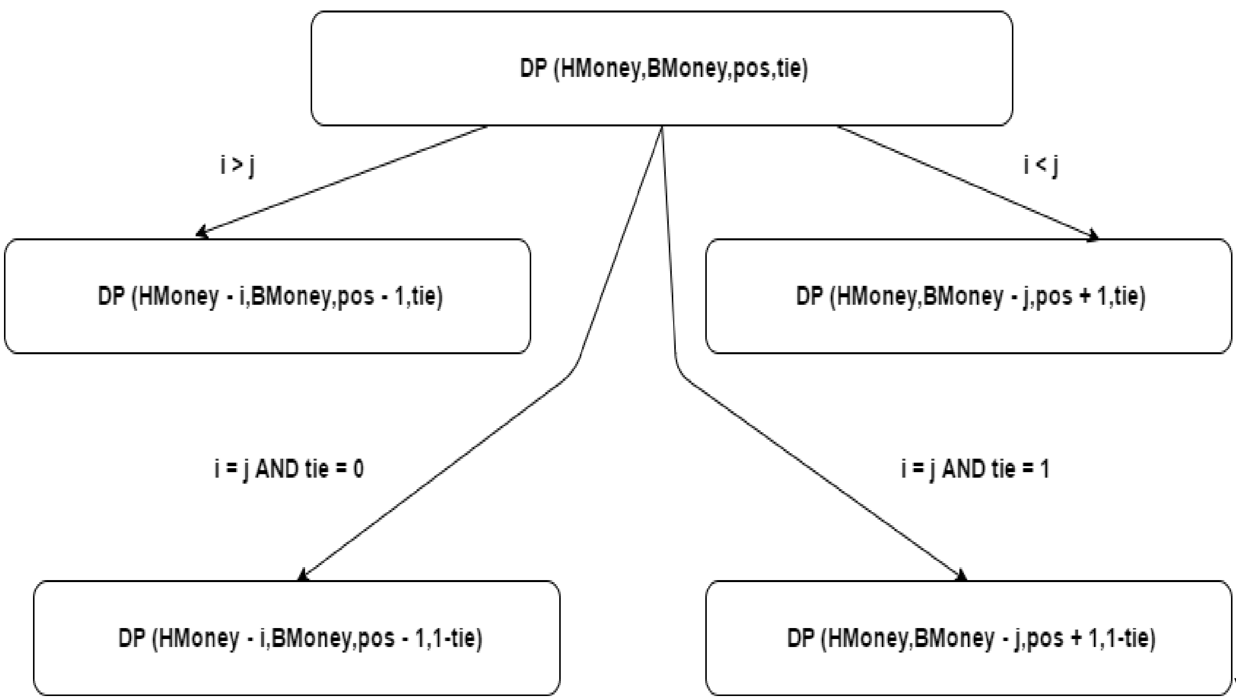

Fig 2.2 Implementation flow chart

The figure denotes the possible transitions of the DP, and the answer for the current state is calculated by taking the maximum of all reachable adjacent states. Since every bid is equi-probable, the probability of each state transition is equal to 1/(HMoney * BMoney) . We define the base cases of our DP recurrence as :

- HMoney = 0, Bot win with probability 1
- BMoney = 0, Bot wins with probability 0
- Pos = 0, Bot wins with probability 0
- Pos = 10, Bot wins with probability 1

Once we have calculated the probability for all states, we will choose the most optimal bid for the bot to increase it's chances of winning.

3.3 Finding the optimal bid

We are now interested in finding an optimal bid for the current state of the game. At first, we find the maximum probability of all the possible visitable states. Among all the bids that have this maximum probability, we choose the least amount. We always bid an amount greater than 1 since bidding an amount equal to 1 is of no use unless its the only possible move. This is because our opponent will atleast bid 1 and hence we would be most probably wasting a chance to move the bottle towards our goal.

Pseudo Code

input: previous moves of Human and Bot,
current position of scotch

output: optimal bid by the bot

```
func preprocess(HumanMoves,BotMoves,moves):
        Human = 100,Bot = 100,Pos = 5,Draw = 0
        for each i in (1,moves):
                if HumanMoves[i] > BotMoves[i]:
                        Human -= HumanMoves[i]
                        Pos -= 1
                elif BotMoves[i] > HumanMoves[i]:
                        Bot -= BotMoves[i]
```

```
                Pos += 1
            else:
                if Draw == 0:
                    Human -= HumanMoves[i]
                    Pos -= 1
                else:
                    Bot -= BotMoves[i]
                    Pos += 1
            endif
        endfor
        return tuple(Human,Bot,Pos,Draw)
end func
func DP(HMoney,BMoney,Pos,Tie):
        if Pos == 0:
            return 0
        elif Pos == 10:
            return 1
        elif HMoney == 0:
            return 1
        elif BMoney == 0:
            return 0
        endif
        val = 0
        for i in range(1,HMoney):
            for j in range(1,BMoney):
                tempPos = Pos
                tempTie = Tie
                if i == j:
                    if Tie == 1:
                        tempPos += 1
                    else:
```

```
                    tempPos -= 1
                tempTie = 1 - tempTie
            elif i > j:
                tempPos -= 1
            else:
                tempPos += 1
            endif
            if tempPos > pos:
                val = max(val, DP(HMoney,BMoney-j,tempPos,tempTie))
            else:
                val = max(val, DP(HMoney-i,BMoney,tempPos,tempTie))
            endif
        endfor
    endfor
    return val/(BMoney * HMoney)
end func
func bid(HumanMoves,BotMoves,moves):
    state = preprocess(HumanMoves,BotMoves,moves)
    bestProb = 0, bestBid = 1
    for i in range(1,BMoney):
        if bestProb < DP(HMoney,BMoney-1,Pos+1,Draw):
            bestProb = DP(HMoney,BMoney-1,Pos+1,Draw)
            bestBid = i
        end if
    end for
    return bestBid
end func
```

4.0 RESULT AND ANALYSIS

The result of the game will be in favour or against the player. Both the outcome of game for a player has been explained below.

Case 1: Human vs Bot and human is a winner.

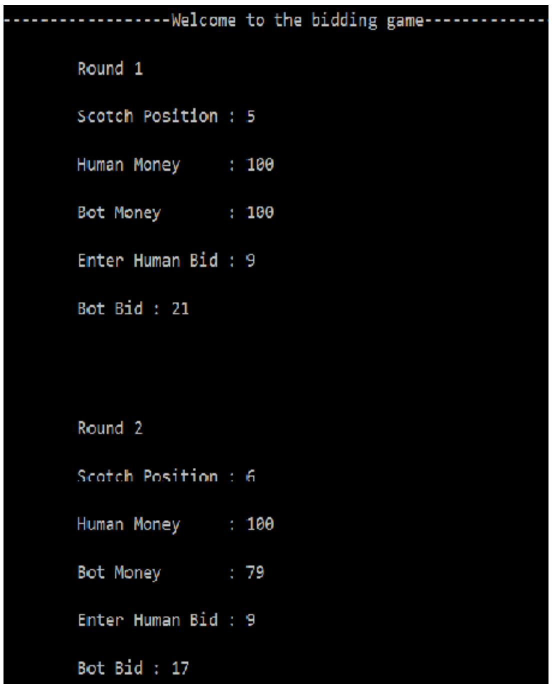

Fig 4.1

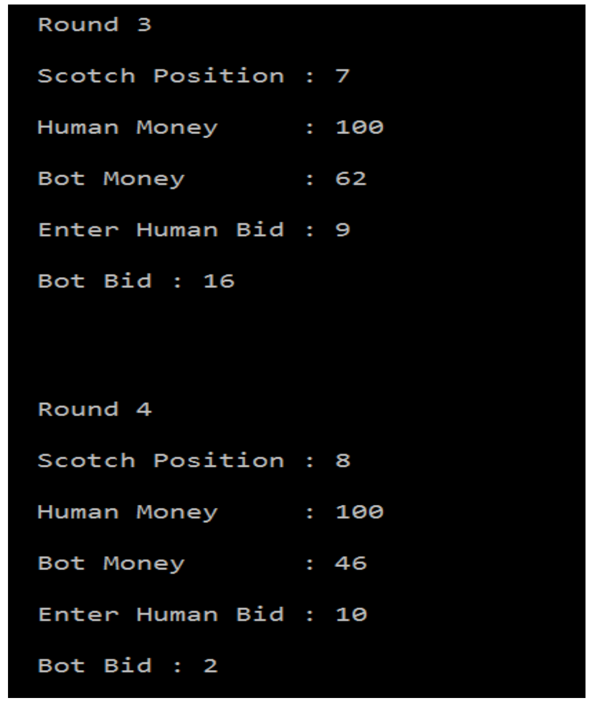

Fig 4.2

Round 1: Human and bot have 100$ respectively, as game starts human will bid certain amount of money in this case human bid 20 and bot bid 21 Fig 4.1 .As its clear that bot bid is higher so he won and that 21$ is reduced from his total money and scotch bottle will be shifted towards bot. As the aim of the game is clearly mentioned if bot have to win it should bid in such a fashion that he should bring the bottle from position 5 to position 10 and human should bring it at position 0 from position 5.

In the similar fashion human and bot continue to bid in each round as shown in Fig 4.1, Fig 4.2, Fig 4.3, Fig 4.4, Fig 4.5, Fig 4.6, Fig 4.7 .

```
Round 5
Scotch Position : 7
Human Money     : 90
Bot Money       : 46
Enter Human Bid : 9
Bot Bid : 13


Round 6
Scotch Position : 8
Human Money     : 90
Bot Money       : 33
Enter Human Bid : 9
Bot Bid : 6
```

Fig 4.3

```
Round 7
Scotch Position : 7
Human Money     : 81
Bot Money       : 33
Enter Human Bid : 9
Bot Bid : 5


Round 8
Scotch Position : 6
Human Money     : 72
Bot Money       : 33
Enter Human Bid : 9
Bot Bid : 5
```

Fig 4.4

```
Round 9
Scotch Position : 5
Human Money     : 63
Bot Money       : 33
Enter Human Bid : 9
Bot Bid : 8


Round 10
Scotch Position : 4
Human Money     : 54
Bot Money       : 33
Enter Human Bid : 9
Bot Bid : 10
```

Fig 4.5

```
Round 11
Scotch Position : 5
Human Money     : 54
Bot Money       : 23
Enter Human Bid : 7
Bot Bid : 2


Round 12
Scotch Position : 4
Human Money     : 47
Bot Money       : 23
Enter Human Bid : 6
Bot Bid : 2
```

Fig 4.6

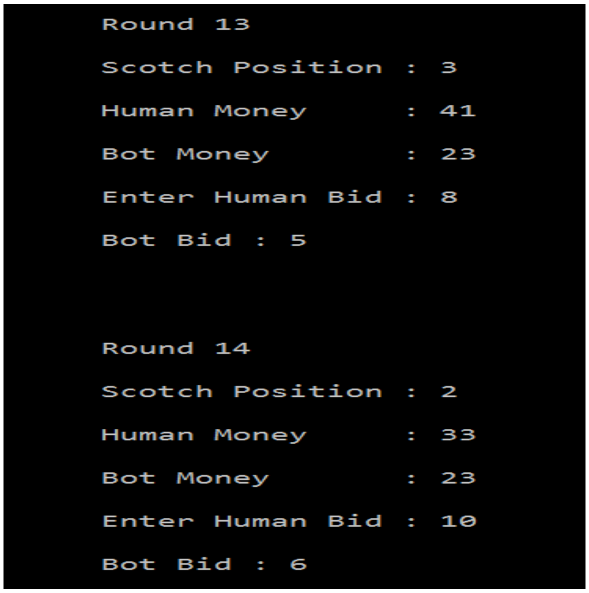

Fig 4.7

Fig 4.8

Round 15: Our game is in round 15 now which means in this case 15 rounds will be played between bot and human and round 15 is the deciding round where human wins the game as , so finally scotch bottle is at position 0 which makes human the winner of this game as shown in Fig 4.8.

Case 2 : Human vs Bot and Human lost the game.

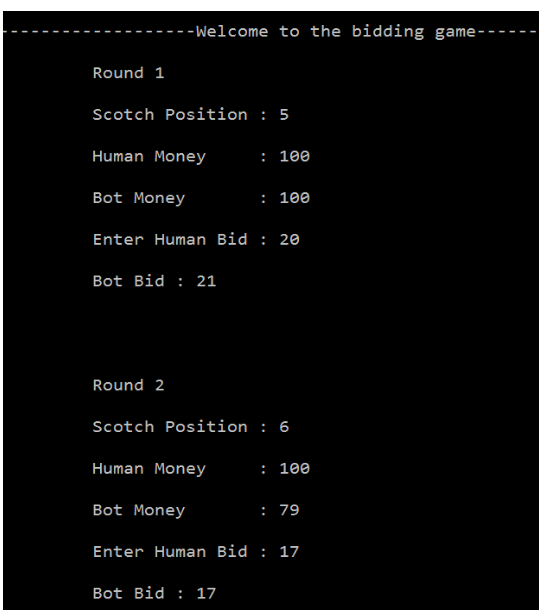

Fig 4.9

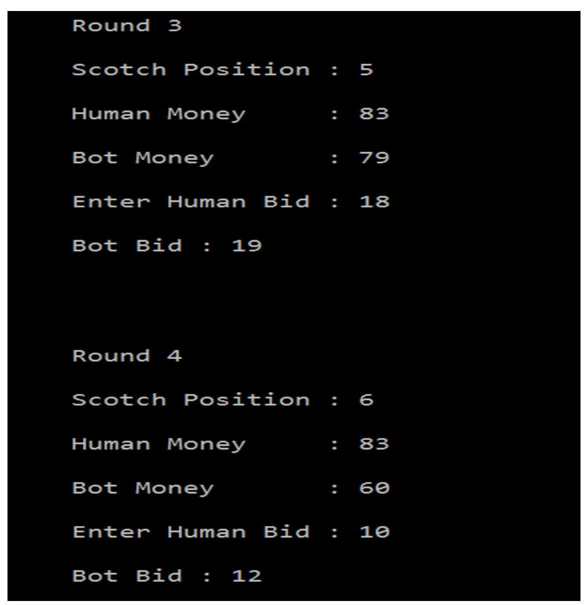

Fig 4.10

Round 1: Human and bot have 100$ respectively, as game starts human will bid certain amount of money in this case human bid 20 and bot bid 21 Fig 4.9 .As its clear that bot bid is higher so he won and that 21$ is reduced from his total money and scotch bottle will be shifted towards bot. As the aim of the game is clearly mentioned if bot have to win it should bid in such a fashion that he should bring the bottle from position 5 to position 10 and human should bring it at position 0 from position 5.

In the similar fashion human and bot continue to bid in each round as shown in Fig 4.9, Fig 4.10, Fig 4.11, Fig 4.12, Fig 4.13, Fig 4.14, Fig 4.15.

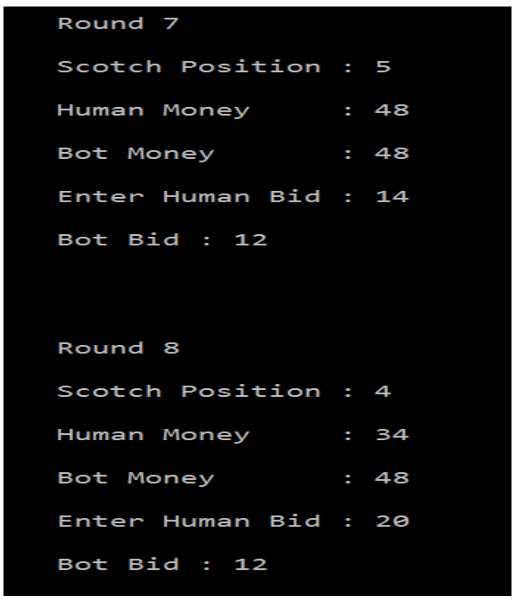

Fig 4.11

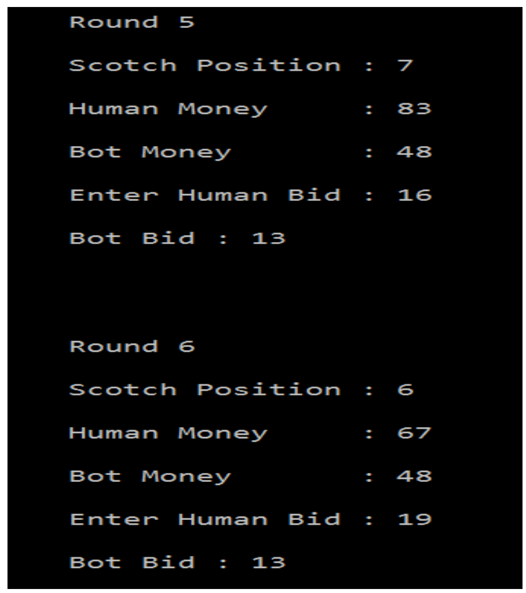

Fig 4.12

```
Round 9
Scotch Position : 3
Human Money     : 14
Bot Money       : 48
Enter Human Bid : 10
Bot Bid : 10


Round 10
Scotch Position : 4
Human Money     : 14
Bot Money       : 38
Enter Human Bid : 10
Bot Bid : 11
```

Fig 4.13

```
Round 11
Scotch Position : 5
Human Money     : 14
Bot Money       : 27
Enter Human Bid : 14
Bot Bid : 5


Round 12
Scotch Position : 4
Human Money     : 0
Bot Money       : 27
Enter Human Bid : 0
Incorrect Bid By Human. Bot Wins.
```

Fig 4.14

Round 12: Our game is in round 12 now which means in this case 12 rounds will be played between bot and human and round 12 is the deciding round where bot wins the game , so finally scotch bottle is at position 10 which makes bot the winner of this game as shown in Fig4.14

S.NO	Opponent's	Games Played	Won	Lost	Tie	Win %
1	Werxzy	2	2	0	0	100%
2	mellinge	2	1	1	0	50%
3	Jin	2	1	0	1	50%
4	Somanyrobts	2	2	0	0	100%
5	legendryartmies	2	1	1	0	50%
6	Hibernicus	2	2	0	0	100%
7	Dustkicker	2	1	1	0	50%
8	Galvez	2	1	1	0	50%
9	Diego_93	2	0	2	0	0%
10	Mharris717	2	0	0	2	0%

Table 4.1 Leader Board Table

Overall Win Percentage is 55%

The above leaderboard is obtained while playing against other bot's across world on HackerRank[02].According to our heuristic one should not always win the game nor one should always loss.If u see above leaderboard in which we played with 10 players our overall win percentage55% which shows the efficiency of our code and heuristic.

Precision Table using Algorithm

Game played	Algorithm precision(score)
Game 1	35.2
Game 2	37.3
Game 3	40.1
Game 4	43.4

Table 4.2

Precision Table using Bruteforce

Game played	Bruteforce precision(score)
Game 1	5.2

Game 2	7.0
Game 3	11.0
Game 4	17.8

Table 4.3

4.1 Inference from tables

Precision tables given above compares the two technique of bidding .The two techniques are Random bidding another one is using optimal bidding method which is explained in section in section 3.2 and 3.3.

The random bidding method neither follow any algorithm nor uses probability to calculate the bid for the system. Its just a random function which generate bids between 1 to 20 while keeping track of money left with bot , if u have money you got your bid it does not care whether it is optimal or not . In simple words we can say that its very similar to trying our luck in a game of throwing dice. The chance of system winning the game is very less which is already shown in precision table 4.2.

The function for brute force bidding technique is as follow

```
if( BMoney > 0)

Bid=rand()%21;

BMoney = Bmoney - bid;
```

5.0 CONCLUSION

We have shown various charts in our previous section 4.1. The statement was very clear, i.e the correctness of our heuristic .From the first chart we concluded the winning % of a player and that was something around 55% which means we have a bidding game that is fair enough for both the sides, i.e both players are equally likely to win. While other chart highlighted the accuracy of our game.

Another important thing that we concluded is using optimal bidding Algorithm is better than using random function.

REFERENCES

[01] Andrew J. Lazarus, Daniel E. Loeb, James G. Propp, And Daniel Ullman "Richman Game" David Richman, (1956–1991).

[02] LeaderBoard "The Bidding Game" HackerRank.