

Bayesian Poker Player Development

by Kym McGain

<u>Introduction:</u>	<u>2</u>
<u>Texas Hold'em:</u>	<u>2</u>
<u>Network:</u>	<u>2</u>
<u>Implementation</u>	<u>3</u>
<u>Sandbagging:</u>	<u>3</u>
<u>Goals:</u>	<u>3</u>
<u>Expert's Advice:</u>	<u>3</u>
<u>Basic Technique:</u>	<u>4</u>
<u>Code:</u>	<u>4</u>
<u>Bluffing:</u>	<u>5</u>
<u>Goals:</u>	<u>5</u>
<u>Expert's Advice:</u>	<u>5</u>
<u>Current Technique:</u>	<u>6</u>
<u>Code:</u>	<u>6</u>
<u>Results:</u>	<u>7</u>
<u>Conclusion:</u>	<u>7</u>
<u>References</u>	<u>8</u>

Introduction:

This project is concerned with the further development of the Artificial Intelligence poker player BPP. The aim of this project was to implement a bluffing and sandbagging technique for BPP. Although bluffing already existed for BPP sandbagging would have to be started from scratch. BPP has been developed using the language Python and all my work will be done within this language.

Texas Hold'em:

The game of Texas Hold'em is a modification on traditional poker. For basic rules of poker please visit this link: <http://www.pagat.com/vying/pokerrules.html>

The Hold'em variations are as follows:-

- All players receive two cards.
- One flop, then turn, then river card is dealt in succession to be share by all players. Thus each player has 7 cards to make a hand of 5.
- Left of dealer plays small blind
- Double left of dealer plays big blinds.

The style of Hold'em that BPP plays is fixed bet heads up. This means, that the game is 1 on 1. It also states that a bet is equal to the large blind, and only 3 re-raises are permitted.

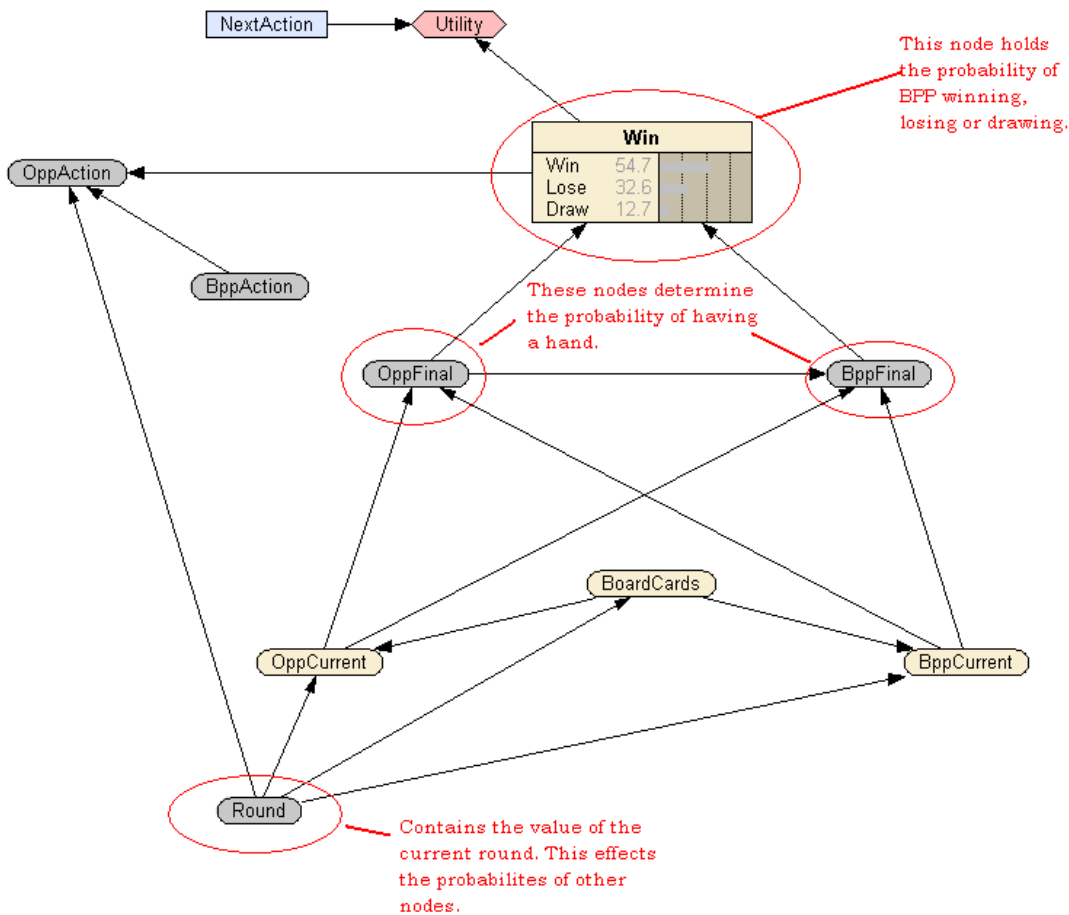
Network:

The way in which BPP makes decisions on what to do each round, is largely based on a probability network. This network is driven by a program called Netica which uses a Bayesian model. A Bayesian network is compiled of different nodes and states which interact with each other. E.g. A node for drink driving can be defined, and a separate node for crashing your car. The probability of a car crash changes with respect to the drink driving node.

Now in this example, the drink driving node would either be true or false, and by setting the value, you are adding what is called findings.

BPP adds findings to it's network about what cards it has, what cards are on the table, etc. And based on these findings, it can extract the probability of winning and then make an informed decision about how to play.

Below, the network is shown with important nodes marked.



Now it's all very well entering in findings by hand, but BPP needs a way to access the network and modify and read it through code. Netica uses a C API library for programs to access it but unfortunately BPP has not been developed with C, but with python. A wrapper for this c library has been written which basically loads the DLL and then points the python functions at the right place in the DLL allowing us to use the Netica library. Being able to use this network is crucial to BPP as Poker itself is largely based on the probability of cards being drawn. Without a way to determine these probabilities BPP would be a fairly shallow minded player.

Implementation

Sandbagging:

Goals:

To successfully implement a sandbagging strategy to allow BPP to win larger amounts of money on the hands in which BPP will dominate.

Expert's Advice:

- A player must have a very strong hand.
- The [free card](#) or cheap card the player is allowing to his opponents must have good possibilities of making them a second-best hand.

- That same free card must have little chance of giving an opponent a better hand or even giving them a [draw](#) to a better hand on the next round with sufficient [pot odds](#) to justify a call.
- The player must believe that he will drive out opponents by showing [aggression](#), but can win a big pot if the opponents stay in the pot.
- The pot must not yet be very large.

Basic Technique:

If the belief of (Win) > 0.6 and a random 60% chance we will switch on sandbagging.

When sandbagging, generate a random number, 75% chance of checking, 25% chance of raising.

This is to randomize the sandbag technique, the consequence of not checking is having BPP act as if there was no sandbag option. This is implied as BPP will raise if it has a high probability of winning under normal circumstances.

This is all done in calcAction() using a global sandbagging variable.

In the river round, sandbagging does not occur allowing BPP to act normally for his hand! We want to raise in this round, most of the money should already be in the pot so folding is no longer an issue. An addition to the basic technique is to not sandbag for that round if the opponents action was to raise or bet. If they are betting, it is unlikely they will fold in the next round so it is okay to raise.

Code:

```

if (round >= THG.numRounds - 2):
    #Reset Sandbagging if in the River or turn
    self.sandbagging = 0

#Kym's Code
#If we are in any round but the last and not already sandbagging
if (self.sandbagging == 0):
    if (round < THG.numRounds-1):
        dout(1, "Deciding whether or not to sandbag...")
        self.env.CompileNet(self.network)

        bppWinNode = self.env.NodeNamed("Win", self.network)
        bppWinBeliefs = self.env.GetNodeBeliefs(bppWinNode)

        #Sandbag if Pr(X=Win)>60% and random 50% chance
        if (bppWinBeliefs[0] > 0.6 and random.random() < 0.5):
            dout(1, "We are SandBagging")
            self.sandbagging = 1
        else:
            dout(1, "We are not sandbagging.")

#Kym's Action
#If we are going to sandbag, 25% chance that we will still raise/bet
if self.sandbagging:
    if (self.Opp.actions[round][0] != 'r' and self.Opp.actions[round][0] != 'b'):
        if random.random() < 0.75:
            action = 'c'
        else:
            action = 'r'

```

Bluffing:

Goals:

To remove the use of the bluff vector and bluff based on evidence and not just random events. Realistically, a bluff vector will still be used, but it will be modified to be more suitable to our other implemented rules.

Expert's Advice:

Lisa Gonzalez

"The advantage to being able to bluff just a couple of players ... well, it's all in the cards. The chances that two other players have a viable poker hand are slimmer, which gives you the upper hand. You want the other two players to build the pot, you want to win maximum amounts and when you've evaluated your players, you've evaluated the chances of a good hand being pulled out by one of them and your own hand in standing ... you're bluff may come through."

My when to bluff scheme is based on the above expert advice. With lack of human interaction, the only judges made in one round must be based on the probabilities of an opponent drawing a good hand.

"One of the biggest secrets is to bluff often but not every hand. This gives your opponents a real task because you're so unpredictable. It's okay to bluff when you have nothing but remember to play it off when you have 'something great'. Being unpredictable and using various bluff tactics can make you the winner. Bluff two or three hands, stop, and play two or three without bluffing. The object is to be unpredictable."

The above statement is also very important in my decision making. While BPP now acts largely on probabilities of the opponents outcome, it also still has a completely random input which allows BPP to act unpredictably. It is very important to not let yourself fall into habits and betting solely on probabilities gives the opponent a great chance of predicting your strategy. If bluffing has a random factor involved, it forces the opponent to either believe you to be bluffing every viable round, which has bad outcomes for them, or to just make a decision each round without learning anything about BPP which completely stops the possibility of falling into traps.

A random outcome here benefits BPP well.

Bill Burton

"Author David Sklansky coined the phrase semi-bluffing. It is a profitable technique that is used instead of an outright bluff. Unlike a bluff when you have nothing, semi-bluffing is done when your hand is not strong enough to win the pot at the time but has the chance of improving to the best hand. If you bet, you are hoping that the other players will fold and you will win the pot without going any further. If you are called then you still have a chance that your hand will improve to be the best hand."

Although my technique does not employ semi bluffing, it is actually a side effect of general bluffing. Often BPP will bluff on a hand which has great potential but is seen at the time as a low hand.

I haven't researched this any further as I feel currently it would interfere with the normal betting scheme, and bluffing would end up occurring at least 50% of the time. I believe that the main bluffing technique can handle semi-bluffing and it doesn't need to be explicitly defined.

Current Technique:

If we are bluffing, then look for most likely higher hand than ours to over compensate our hand and bet higher.

Proposed Changes: Remove all knowledge but the 'flopped' cards and look at the probability of BPP having a good hand. Only bluff if the opponent believes it probable for BPP to have a good hand.

Issues: In order to decide how good the other players hand is, we look at the probabilities of hands based only on what is in the flop. To then compare this with an opponents hand which we have no knowledge of, we are comparing the same probabilities so it will always come out as both being the same.

Method Used:

If the probability of any hand excluding low or medium pair is > 0.15 then it is justified to bluff. Furthermore, if the probability of an opponent raising or betting is > 0.3 then do not bluff. Note: This is not based on their actual actions, BPP will bluff still if they decide to bet, but BPP makes a decision to bluff based on the likelihood of the opponent folding.

A final rule is used, if the opponent is raising/betting in the river then they will not fold, so BPP should stop bluffing. Furthermore it might be advisable for BPP to fold although I have not implemented this and this will be decided by BPP's Bayesian network.

Code:

```
#Don't bluff if we are sandbagging ;)
if not self.sandbagging:
    if not self.bluffingOff:
if not self.bluffing:
    if random.random()<self.bluffVector[round]:
        #Calculate Probabilities of having a good hand from oponents perspective.
        self.env.CompileNet(self.network)
        bppCurrentNode = self.env.NodeNamed("BppFinal", self.network)

        #Clear Node Findings to have an unbiased view of our own hand
        self.env.RetractNodeFindings(bppCurrentNode)

        bppCurrentBeliefs = self.env.GetNodeBeliefs(bppCurrentNode)

        #Find BPP's most probable hand
        #Start at 3 so as to skip the busted hand, low pair and med pair as these
        #are usually the most likely

        #We are only interested in 'better' hands
        for i in range(3, self.env.GetNodeNumberStates(bppCurrentNode)):
            #Pr(Good Hand>0.15) and must not already be on the board.
            #eg. Pr(X)=1.0
            #Using currentBeliefs[i] != 1.0 is possibly not correct,
            #may need to redo this!
            if (bppCurrentBeliefs[i] > 0.15 and bppCurrentBeliefs[i] != 1.0):
                self.bluffing = 1
```

break

```
if self.bluffing:
    #If round is the turn or after
    #Don't bluff if Pr(opponent raising/betting is high)
    if (round >= 2):
        oppActionNode = self.env.NodeNamed("OppAction", self.network)
        self.env.RetractNodeFindings(oppActionNode)
        bppActionBeliefs = self.env.GetNodeBeliefs(oppActionNode)

        #If they are likely to bet or raise then let's not bluff
        if (bppActionBeliefs[3] > 0.3 or bppActionBeliefs[4] > 0.3):
            self.bluffing = 0

    #Finally if opp is betting/raising in the river, don't bluff.
    # It will only lead to a higher loss.
    if (round == THG.numRounds - 1):
        if (self.Opp.actions[round][0] == 'r' or self.Opp.actions[round][0] == 'b'):
            self.bluffing = 0;

if self.bluffing:
    #Set BPP action to the bluff action
    myCalculatedState = self.doBluff(round)
```

Results:

These results are from my version of BPP playing the opponent, version 3.14 of BPP.

```
bpp3_1_4 20:1000 Session: 1 - (<$-100.00 | -0.01 sbu | 0.00% | 474.00)
bpp3_1_4 20:1000 Session: 2 - (<$425.00 | 0.02 sbu | 50.00% | 484.00)
bpp3_1_4 20:1000 Session: 3 - (<$-1170.00 | -0.03 sbu | 33.33% | 486.33)
bpp3_1_4 20:1000 Session: 4 - (<$4050.00 | 0.08 sbu | 50.00% | 490.75)
bpp3_1_4 20:1000 Session: 5 - (<$330.00 | 0.07 sbu | 60.00% | 487.40)
bpp3_1_4 20:1000 Session: 6 - (<$2050.00 | 0.09 sbu | 66.67% | 484.50)
bpp3_1_4 20:1000 Session: 7 - (<$1485.00 | 0.10 sbu | 71.43% | 484.14)
bpp3_1_4 20:1000 Session: 8 - (<$-4405.00 | 0.03 sbu | 62.50% | 480.13)
bpp3_1_4 20:1000 Session: 9 - (<$4425.00 | 0.08 sbu | 66.67% | 483.33)
bpp3_1_4 20:1000 Session: 10 - (<$3750.00 | 0.11 sbu | 70.00% | 485.50)
bpp3_1_4 20:1000 Session: 11 - (<$2830.00 | 0.12 sbu | 72.73% | 486.36)
bpp3_1_4 20:1000 Session: 12 - (<$-525.00 | 0.11 sbu | 66.67% | 485.67)
bpp3_1_4 20:1000 Session: 13 - (<$220.00 | 0.10 sbu | 69.23% | 485.92)
bpp3_1_4 20:1000 Session: 14 - (<$-415.00 | 0.09 sbu | 64.29% | 485.43)
bpp3_1_4 20:1000 Session: 15 - (<$4625.00 | 0.12 sbu | 66.67% | 486.40)
bpp3_1_4 20:1000 Session: 16 - (<$2955.00 | 0.13 sbu | 68.75% | 486.81)
bpp3_1_4 20:1000 Session: 17 - (<$-1050.00 | 0.11 sbu | 64.71% | 486.94)
bpp3_1_4 20:1000 Session: 18 - (<$1275.00 | 0.12 sbu | 66.67% | 486.67)
bpp3_1_4 20:1000 Session: 19 - (<$-4035.00 | 0.09 sbu | 63.16% | 484.95)
bpp3_1_4 20:1000 Session: 20 - (<$-1240.00 | 0.08 sbu | 60.00% | 484.50)
 20 Sessions, 1000 Hands
Sessions won: 60.00% <12>
Sessions won t<19>-value: 0.89
Earnings: 0.08 sbu <$774.00)
Earnings $d: 0.26 <$2562.06)
Earnings t<19>-value: 1.32
Hands won per session: 48.45% <484.50)
Showdowns per session: 56.92% <569.20)
```

Important things to note:

60% of all sessions were won.

Total Earnings: \$15,480

Conclusion:

Judging by BPP's new observed behavior, the new techniques have a large impact on the way BPP plays each round. BPP now plays in a way which is more appropriate for normal games, whether this will improve

playing performance is another thing. From the results above, BPP has definitely improved against itself, but the results are only one case and are not statistically convincing. More testing of BPP would be needed to discover an accurate record.

References

<http://www.blindbetpoker.com/strategy/bluffing-strategy-2.html>

http://casinogambling.about.com/cs/poker/a/bluffing_2.htm

http://en.wikipedia.org/wiki/Slow_play_%28poker%29

<http://www.pagat.com/vying/pokerrules.html>