THE GAMBLER: A BAYESIAN BELIEF NETWORK FOR PLAYING ONLINE POKER (and winning)

by

Nathan Matthews

Temple University, 2005

Introduction

The goal of this project was to develop *Gambler*, an artificial intelligence based on Bayesian Networks (also called Causal Networks, Belief Networks, or Directed Markov Fields). The purpose of the AI is to play on line poker (and win). The method used should apply to many card games or games in general, but for this project the network was constructed specifically for playing Low-Limit Texas Hold'em. As a test of the intelligence of the network, a custom "poker-bot" was written to allow playing with real money on *partypoker.com* and *pokertable.com*. It was shown that this technique produced a net-winning strategy.

Section 1 of this paper outlines the specific structure of the poker rules involved and the important concepts in winning strategy. Section 2 introduces the structure and terminology of Bayesian networks. Section 3 presents the first half of the *Gambler*: the pre-flop network, and Section 4 presents the second half, the post-flop. Section 5 outlines several methods of solving Bayesian networks which were tried, and compares them. Section 6 presents the results of tests of the program, and in Section 7 I give my conclusions and thoughts about future work.

1. Texas Hold'em

Anywhere from two to ten players can play at a table in Low-Limit Hold'em. Every round, a new player is the dealer. The dealer is always the last player to act in a turn. Unlike other card games, all players do not have to ante up to play in the round. The player to the left of the dealer is called the small blind, and this player *does* have to ante up, usually half the minimum bet amount. To the left of this player is the big blind, who has to ante twice that amount.

The game begins with the pre-flop round. When this round begins, the blinds place their bets and then action proceeds clockwise from them until at last the dealer acts. Since the blinds were forced to bet, they then act again (the small blind must call the big blind and any other bets, or fold). Players

cannot check in the pre-flop round. Every round thereafter (post-flop, post-turn, and post-river), the blinds will act first (without being forced to bet) and the dealer will act last.

Every player at the table has two cards they hold and do not show. These are called the hole, or the pocket cards. During the course of the game, cards will be added to the table until five are added. The player's hand is formed by taking any five of the seven cards that player can see; the cards on the table are shared, and any part of them is some portion of each player's hand.

The first round is called pre-flop because no cards are on the table yet. The betting in this round is based purely on the players' hole cards, and by watching other players' actions. After this round, three cards are placed on the table; these are the flop cards. Each player now potentially has a hand, but there are still two cards to come and a player's hand may improve (and so may his opponent's). After the flop, betting starts with the small blind and proceeds clockwise. The small blind may check, and so may any player who is checked to. If there is a bet on the table, you must either call, raise or fold.

After the post-flop betting is done, one more card (called the turn card) is placed, and betting repeats. Last, the final card (called the river card) is placed, making a full seven visible cards per remaining player. The final round of betting is done, and then the winner is determined by the strength of his hand. The hands are, by increasing order of strength, a high card (twos are low, aces high), a pair (two cards of equal rank), two pair (two pairs of different rank), three of a kind (three cards of equal rank), straight (five cards of sequential rank), flush (five cards of the same suit), full house (three of one rank, and two of another), four of a kind (four cards of equal rank), and straight flush (give cards of equal suit and sequential rank). It is well known that a royal flush is the best hand (T,J,Q,K,A of equal suit), but this is simply the straight flush with the highest cards.

Texas Hold'em, Limit and no-Limit, is not an easy game to play well. There are many intricate concepts to balance at once, and often they are contradictory. There are general concepts, however, that always hold true and which are good to keep in mind as a framework to decide the overall

structure of the Bayesian network. So, in summary, there are three major factors a player considers to determine his standing at the table: what cards his opponents likely hold (determined by watching their actions), hands the player could draw in subsequent rounds and their likelihood (called pot odds), and the amount of the pot compared to the amount of the bet the player must put in. After the player determines what his standing at the table is likely to be, he determines whether it is worth putting money into the pot (and in certain circumstances, will decide to follow some more complicated strategy, like isolation, bluffing, or checking to known loose players to increase the pot).

Now, in deciding the general structure of the Bayesian network(s), consider the rationale for using a belief network to play poker in the first place. Why is this mechanism well suited to the problem? First, one of the major niches in AI filled by Bayesian networks is reasoning with uncertainty. Bayesian networks are able to produce good guesses even when the full state of the problem space is not known, and when only partial causal information is present to deal with it. Uncertainty is the main characteristic of poker, and the human mind already performs simple probabilistic leaps when playing the game which translate into causal information.

Second, while there is a great deal of formal poker strategy in books and tutorials, there are too many rules of too small scope: resources give advice for specific situations in poker, or for overall strategies, but not a structure which is cohesive enough for a decision-based computer program. It is difficult to take the thousands of small rules given for every poker hand, and unify them into one algorithm. Furthermore, even poker experts admit that the general, well-founded rules for winning poker are sometimes contradictory and must be weighed intuitively. Intuition is not something that software is normally very good at, but Bayesian networks can overcome that difficulty because they are probabilistic.

Third, and most convincingly, given a player is able to decide correctly what to do if he knows his standing at the table, it is more difficult to guess what position the opponents are in. Knowing what position the opponents are in is half the game, and yet this step, outside of a belief network, must be a

pure guess. However, if the player can make the assumption that his opponents are rational players operating in much the same way he does, he can "reverse-engineer" their *actions* at the poker table into the *reasons* they were likely to have made those choices, using the same rationalizations the player would have. It is possible for human players to do this operation intuitively, but not with great accuracy, meaning human players must make more conservative estimates than are necessary. This kind of "reverse inference" is very easy for a Bayesian network to do; simply reverse which variables of the network are observed and which are queried, terms which are presented in the following section.

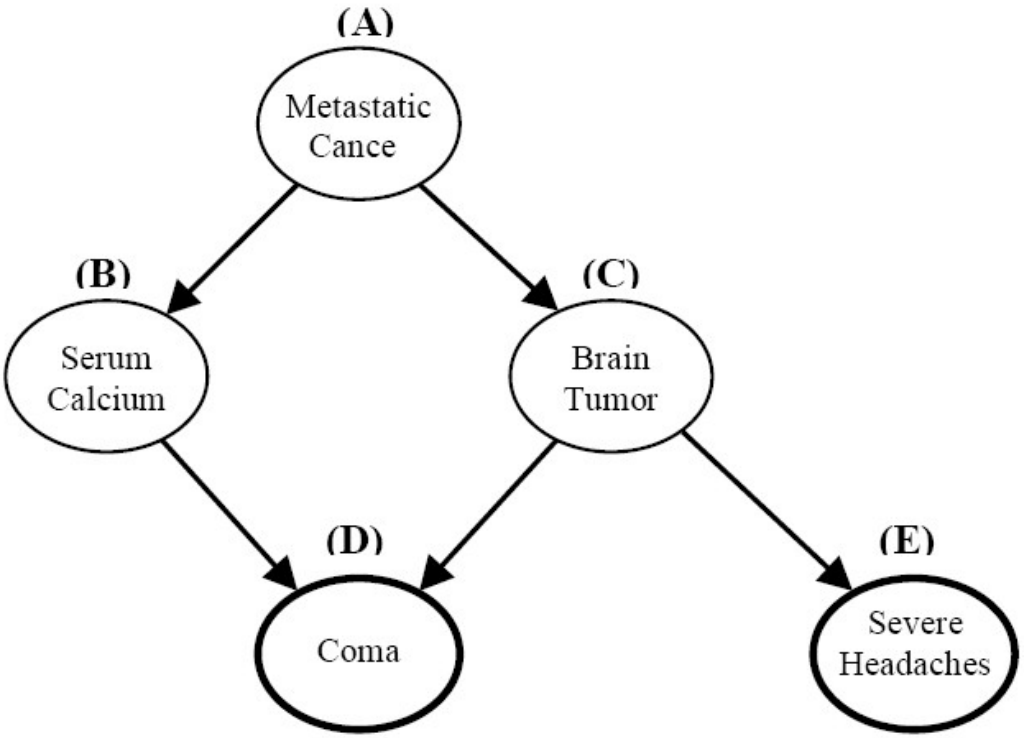
2. Bayesian Networks

The idea for a Bayesian network arose out of the need for reasoning with uncertainty in AI. In dealing with a problem (in this case, a game), the AI must use the known state of the game (it's variables) to determine some action. But variables are not always things which can be observed directly: in poker, the hole cards of the opponents are unknown, and yet have some particular state. On the other hand, if those cards *were* known, a player would know exactly what to do: this is because the rules of poker, given perfect information, can be reduced to a set of "if/then" hypotheses; e.g., if the player holds an ace pair, and the opponent holds a ten pair, raise the bet. Therefore, a Bayesian network consists of two parts.

The qualitative part of the network consists of variables and causal links. The variables represent some aspect of the problem; for instance, the player's cards, the table cards, the player's position, whether any players have raised a bet, etc. Each variable can be in one of a set of discrete states; for instance, the "position" variable, indicating at what order in play the player gets to act, could be small_blind, big_blind, early, middle, etc. In the evaluation of the network, some variables are observed by the AI (in poker, the player). For instance, position at the table is always known. For observed variables, their state is fixed. For unknown variables, inference algorithms will try to determine the state of the variable, but not with complete certainty. Instead, the variable will have

some probability of being in each state, as long as the sum of these probabilities is 100%.

The causal links of the network, with the variables as nodes, can (generally) be represented as a directed acyclic graph. That is, links between variables have direction, and no cycles can be formed among nodes. A link from variable "raining" to variable "yard-wet" would indicate that the yard being wet is conditional on whether it has rained. Below is an example network which explains these concepts:



$\Pr(A)$	
a_1	0.2
a_2	0.8

$\Pr(B\|A)$	a_1	a_2
b_1	0.8	0.2
b_2	0.2	0.8

$\Pr(C\|A)$	a_1	a_2
c_1	0.2	0.05
c_2	0.8	0.95

$\Pr(E\|C)$	c_1	c_2
e_1	0.8	0.6
e_2	0.2	0.4

$\Pr(D\|B,C)$	b_1		b_2	
	c_1	c_2	c_1	c_2
d_1	0.80	0.80	0.80	0.05
d_2	0.20	0.20	0.20	0.95

Fig. 1 – the "Coma" network [Cheng, 2001]

In this network we see unknown variables A through E, each of which has two possible states (which indicate that named symptom/disease is present or absent, respectively). The nodes of the graph and the parent-child links between them make up the qualitative part of the network.

The tables in the diagram are the quantitative part. They give two types of probability distributions: prior distributions, for variables with no conditioning parents (in this case, only variable A), and conditional distributions, for variables which have a probability density conditional on the state(s) of their parent(s).

Let Pr(A) represent the prior probability of the variable A being in state a1. That is, given no evidence, A is in state a1 with 20% probability and in state a2 with 80% probability. Pr(B | A) is the conditional probability Pr(B), given A is true. Therefore if A is in state a1, B has an 80% chance of being in state b1; if A equals a2, it has only a 20% chance. Pr(D | B,C) indicates the conditional probability of D given both its conditioning parents (since D has two converging arrows in the graph, from B and C). D equals d1 with 80% probability in all cases except when B is b2 and C is c2, in which case D is only d1 with 5% probability. Note that for each combination of states of B and C, the sum of d1 and d2 must equal 100% to form a normalized probability density.

Now let all variables of the network be represented by the set **X**. Some subset of these variables are observed in a known state (call these "evidence" variables **E**), and some are to be queried (we want to find out the distribution of the states in those variables). Call the set of query variables **Q.** So, what we want to determine is Pr(**Q** | **E**), the joint probability of the query variables given the evidence. By the definition of probability, this is equal to Pr(**Q,E)** ÷ Pr(**E)**. We can use probability to obtain Pr(**Q,E**) from Pr(**X**) by summing out all non-evidence, non-observed variables. Pr(**X**) is called the total joint probability, and it is very difficult to calculate normally. If there are N variables in the network, each with B states, the order of magnitude in size of the joint probability is B to the power of N, giving exponential space and time requirements for the solution. Therefore, there are several mechanisms which will be described later, which can give both exact and approximate solutions.

The meaning of the total joint probability of the network is that it indicates, for each possible combination of the states of all network variables, the probability of that configuration occurring. Because of the Markov Independence rule, the joint probability can be calculated by multiplying the probability distributions of each variable conditional on its parents. This is because each variable can appear only once as a conditioning variable, and because of the notion of d-separation. If **X** is the set of all variables in the network, and Pa(**X)** represents the parent variables of **X**, then the joint probability of **X** is:

$$\Pr(\mathbf{X}) = \prod_{i=1}^{n} \Pr(X_i | Pa(X_i))$$

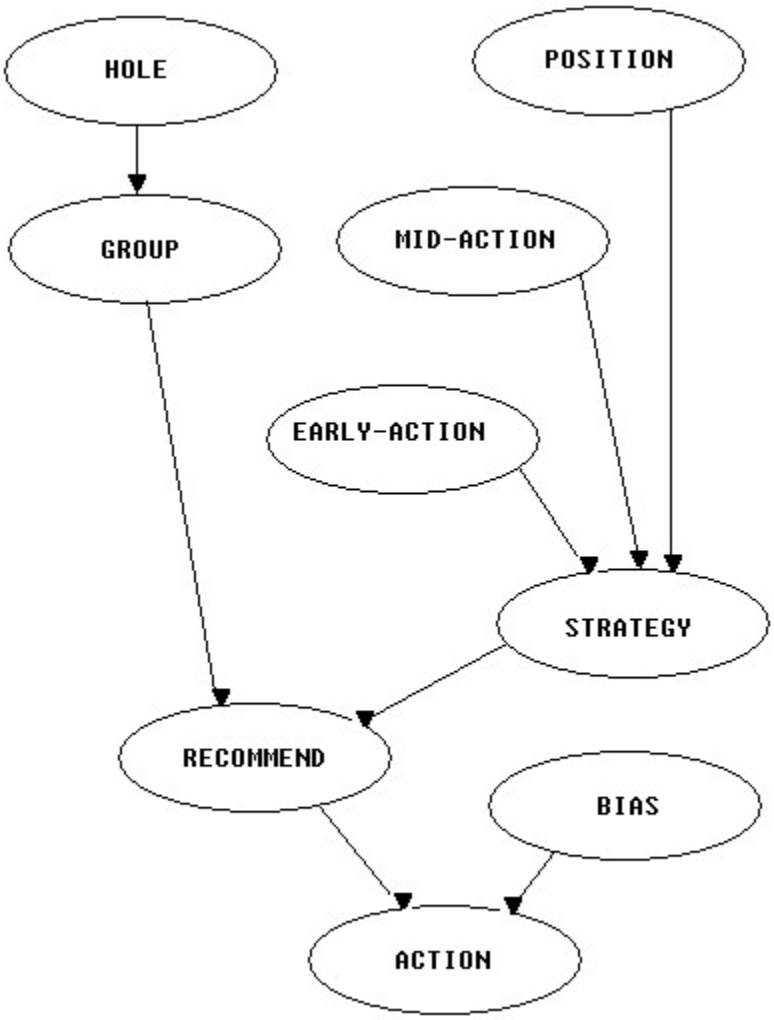
The observed variables are in a known state and are summed out of the distribution after finding the joint probability, to find total probability conditional on the evidence. In practice, since the resulting distribution is huge for a moderate number of variables, in order to find out information about the posterior distribution of a "query" variable (its marginal probability given its conditioning variables), alternative inference methods exist to limit the number of multiplications in the process. Algorithms such as bucket tree, junction tree, conditional independence, and recursive decombination apply this method. However, the problem has been proved NP-hard, and for large networks is still prohibitively slow. In the following sections, the structure of the Bayesian networks in Gambler are presented, and in Section 5 the tested solution methods are given and compared.

3. The Pre-Flop Network

When adapting the game of poker to Bayesian Networks, it is clear the the pre-flop stage is distinct from the three post-flop stages. For one, early position players (those to act first) know almost nothing about their opponents in this round, and must react strictly according to their hole cards. Also, there is little information for any player about draw probabilities, making the post-flop computation of

pot odds impractical. Players must also in general play much tighter pre-flop (that is, take less risk, as opposed to playing loose). The three post-flop rounds are almost identical in the mechanics of deciding actions, differing only in probabilities and strategies. They are all represented by the same network.

The pre-flop network was designed as follows.



These variables, the hole, group, position, early-action, mid-action, strategy, recommend, bias, and action, make up the entirety of a pre-flop round. The hole variable has states corresponding to the values of all card combinations; there are 13 pairs, and then combinations of non-paired cards which can be either suited (having the same suit) or unsuited. Each of these combinations has a power rating which was determined experimentally. AA (an Ace pair) is the highest (rank 40), and 72u (deuce and seven unsuited) is the lowest, with rank 0. The prior distribution (the probability of each combination, drawn randomly) initializes each pair to .45%, each unsuited combination to .9%, and each suited combination to .3%.

The group variable is determined based on the hole cards. It separates these cards into six groups of different power, each of which is either usable or not usable in some scenario at the table. These groups were determined by finding break-even points for power ratings given specific scenarios and positions at the table, and were tested experimentally. They conform to an intuitive sense of which group of cards will consistently beat a lower group of cards.

The position variable indicates the player's relative position at the table (the order they act in). It can be one of under-the-gun, small-blind, big-blind, early, middle, or late.

Early-action and mid-action both have the states fold, call, and raise (there are no bet and check states, because the blinds always bet and nobody can check pre-flop). For different groups of player positions (earlier and later), these represent the most aggressive action taken in that group of players. If any early-position player raised, the variable early-action will be in state 'raised'. The same goes for mid-action.

The strategy variable is the most important in the network. Its states and density function were designed and tuned based on a large volume of expert poker advice. Based on the player's position at the table, and the actions of the opponents, the strategy variable determines how safe the player should feel at the table, and what action should be taken if it's safe not to fold (calling or raising). In Low-

Limit Hold'em, a player almost always raises pre-flop because the best winning strategy is to play tight-aggressive, which means folding most hands but raising and re-raising really good hands. The variable was designed with this strategy in mind; in early position, all but very good hands indicate a flop. In later position, as long as there wasn't excessive aggression, worse hands become playable. The states of this node are S1-S16. Each will represent, in another variable, which group the player's hole cards must fall in not to fold, and if not folding, what action to take.

The recommend variable has states fold, call, and raise. It is conditional on strategy and the player's power group. If a strategy allows the group, the action indicated by the strategy is the state recommended; otherwise, it recommends folding.

The bias variable has states loose, normal, and tight. This variable is always an observed variable and is a tunable property of the AI. It indicates that the players at the table are known to play looser or tighter than is probabilistically justified.

Finally, the action node takes the recommended action and modifies it based on bias. If the bias is tight, the action will sometimes be to call when raising is recommended, or to fold when calling is recommended, and the opposite for a loose bias.

As indicated earlier, the power of a Bayesian network is that it can perform inference from either prior evidence, or from “posterior” evidence. So when it is time for the player to act, the hole variable is observed, as are the position, early-action, mid-action, and bias variables. The queried variable is the action variable. Because inference proceeds from all known variables, one of the states fold, call, or raise will have 100% probability, and this is the action which should be taken.

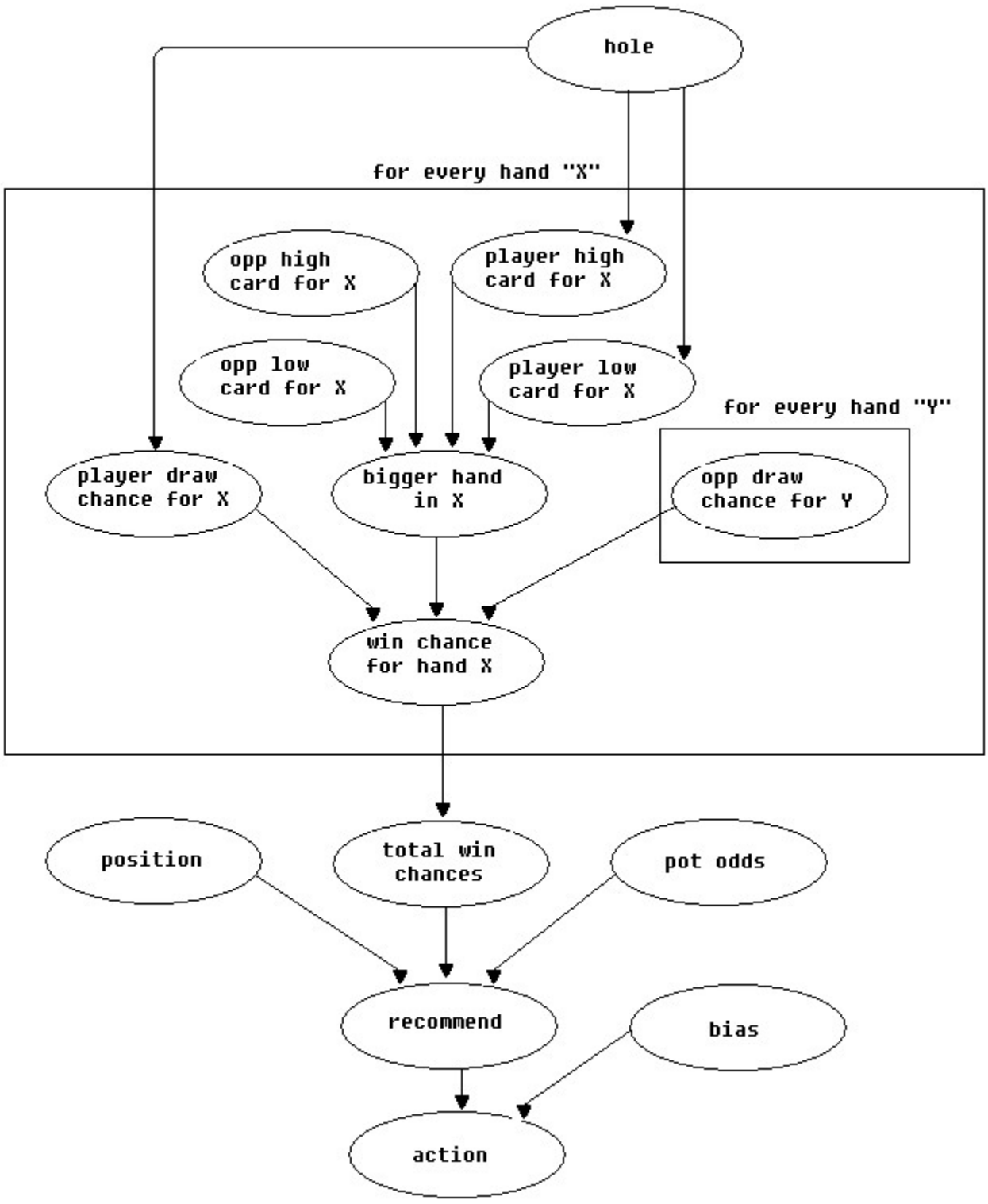
However, for each opponent who acts, the network can be reversed; the action is an observed variable (the player is observing the opponent's action), and the hole variable is queried to determine the distribution of cards which would have caused that opponent to act in that way. This distribution will be used later in the post-flop rounds, but it is the real crux of the network's power that given a position, and table aggression, and the opponent's action, a full distribution of their cards can be found.

This is something that human players can do only approximately, and so they must be conservative with their estimation and will win less over all.

If betting proceeds around to the big blind and that player raises, betting will continue around. In this case, the opponents' projected hole distributions are multiplied to determine the probability, for each hole combination, that *some* player has those cards. Then, multiply each of these values whose combination has a higher power rating than the player's, to get the total probability that some opponent's hole dominates the player's hole. If the win ratio is at least equal to the ration of the pot odds (the amount of the pot versus the bet amount), the player will call this bet, or else fold. Domination by power rating is an approximate form of what happens in post-flop rounds, because without a flop it is impossible to calculate real pot odds.

4. The Post-Flop Network

The post-flop network is different from pre-flop primarily in that it relies more directly on the estimated probability of winning certain hands, and less on intuitive understanding of aggression (which is still a factor, but indirectly). Despite having a more clear-cut strategy, this network's structure is actually more complicated. Each possible hand must have an independent draw probability, the maximum opponent's draw probability, the probability of dominating each opponent hand with each user hand, resulting in the probability of winning on every possible hand, producing the total probability of being the strongest draw on the table. Finally, these win chances are compared to the pot odds to determine whether to call or raise any bet. The network looks like this:


hole
for every hand "X"
opp high card for X
player high card for X
opp low card for X
player low card for X
for every hand "Y"
player draw chance for X
bigger hand in X
opp draw chance for Y
win chance for hand X
position
total win chances
pot odds
recommend
bias
action

Notice that variables within boxes are repeated for every possible hand; links in and out of the boxes (i.e, from “bigger” and to “win odds” are duplicated, and each “opp draws” is cross-linked to each “win hand”).

Before the player's turn, some initial calculations are performed. Three cards on the table are known, and the player's hole cards are known. The operation known as “counting outs” (seeing how many un-observed cards could complete each hand) is performed to intialize the “player draws hand” variables conditional on the hole cards. This conditional distribution is not fixed, but must be changed whenever the table cards change, because the draw chances change. Much the same operation is performed for each opponent using the hole distributions projected from the pre-flop round. The result is a set of draw probabilities for each opponent and hand. These densities are added to the “scare pool”, and then are multiplied to determine the probability that *any* opponent will draw each hand, and these values are used to initialize the “opponent draws hand” variables. Along with the draw chances, the high and low cards for each hand are calculated (some hands, such as a high card, only have one of these two).

The “bigger hand for X” variable will be True if the high/low cards of the player beat the high/low cards of the opponent, and false otherwise. Finally, the “win hand X” variable will be true in the case where the player draws that hand, and the opponent either does not draw any higher hand, or if the opponent draws the same hand, that the “bigger X” variable is true. Otherwise, it is false. The distribution of every hand X variable indicates the chance that the player will dominate the table with that given hand, once zero, one, or two more cards are added to the table (depending on the round).

The “win” variable combines each hand's win chances. If any hand can dominate the table, the win variable is true. The density of this variable indicates the total probability that the player will end up having the highest hand in the table.

Once these variables are combined to form the total probability of dominating the table, the next two evidence nodes, stakes and position, determine the recommended action. In general, in the

post-flop rounds, folding occurs when the size of the pot does not warrant the risk of a bet. For instance, if the pot is $40 and the bet is $5, the pot odds are 40:5 or 8:1. Therefore, the player need only win this pot one out of every eight times to break even, or more to profit. Therefore, a win probability of *less* than 1 in 8 indicates a fold. In addition, the position variable indicates greater uncertainty in early position because no action has been seen yet, and riskier bets are avoided. The combined probability of making a profit on this round (in addition to strategy relating to the position of the player) will determine the recommended action, which is modified by bias as before to produce the actual action.

The method is only slightly different for opponents. Each opponent's "opp draw" variables must be initialized as if they had observed the player as any other opponent, by adding the player's draw probabilities to the scare pool and removing their own, then repeating the draw chances calculations. Then, as before, the hole variable is queried while the action variable is observed. After this query is done, the projected hole density for this opponent is used as an observed variable to query the draw probabilities for that opponent, given the outs distribution for the table. These draw probabilities then replace that opponent's entry in the scare pool, to be re-evaluated by every subsequent player in the round. In this way, players' actions affect the projected distributions of opponents' hole cards throughout the post-flop rounds.

5. Solving the Network

The structure of these two networks guided the search for an appropriate algorithm to query the network during game play. First, it must be easy to have evidence nodes either at the "top" or the "bottom" of the network, and to perform equally well. Second, since on line poker usually proceeds very quickly (a second for most opponents to act), the network evaluation must be very fast.

There are a variety of simulations and approximations to solving Bayesian networks which produce adequately precise results (under certain conditions). For an initial attempt, the poker

networks were approximated using a quasi-Monte Carlo technique called Logic Sampling using Markov Blankets. In this method, each non-evidence variable is initialized to a random state. Then for a certain number of iterations, each variable in topological order evaluates is posterior distribution based on its Markov blanket, that is, the variables in the network which make it conditionally independent from the rest of the graph: these are its parents, children, and childrens' parents. Mathematically, this is equal to the distribution:

$$Pr(X) = Pr(X \mid Pa(X)) \times Pr(Ch(X) \mid X, Sp(X))$$

Where Pa(X) are the parents of X, Ch(X) are the children of X, and Sp(X) are the "spouses" of X (the parents of X's children). The variable then resets itself to a new state randomly, such that the outcome is proportional to the calculated distribution. Each iteration of this method is a sample of the total joint probability, and because the variable distributions are used to select each successive sample (making it a quasi-Monte Carlo method), the result is an average distribution of states for each variable, given the fixed evidence nodes.

As is noted in the literature and confirmed by tests, this simple method of logic sampling suffers from a lack of adequate sample distribution. Evidence with very low probability in the network will tend to never be sampled, skewing the resulting joint distributions. In the case of both networks, every single state in the hole variable has a very low probability, making this method highly inaccurate in practice. In tests of simple networks of only five nodes which included low probability conditional distributions, error of up to 1% with 10,000 iterations was observed. While this may be very low in general use of Bayesian networks, for poker it is significant enough to worry about.

There are many methods available which can help correct the sampling problem: stratified sampling, importance weighting, and latin-hypercube sampling, to name just some. Latin hypercube sampling, in particular, has shown to improve the sampling accuracy of other schemes. In the case of poker, latin hypercube sampling was implemented but because the networks are relatively small, did not significantly drop the error below .5-.9%.

Instead of approximation, a technique exists to calculate the probability Pr(**X, E**) with a smaller number of multiplications (and space) than is necessary when getting a total joint probability. The method chosen, Generalized Variable Elimination with Bucket Trees, was the simplest to implement and the most flexible because it can be generalized to query every node of the network with equal ease.

In this method, some ordering of non-observed, non-evidence variables is chosen given a heuristic function (the chosen function was the minimum neighborhood weight function). Initially, all network densities are added to a pool. Then, in the order of the heuristically-chosen variables, every density containing that variable is extracted and multiplied together in a "bucket" which now contain those densities. That variable is then summed out of that distribution, thus eliminating the variable from the network. This new density is then added back to the pool. Variables with the most multiplications are eliminated from the network first. When each non-query variable is eliminated, the remaining densities are multiplied together and the evidence variables summed out, to obtain the probability Pr(**Q | E**).

A further method presented by Cozman further generalizes this process to be able to then query any variable in the network, but this turned out to be unnecessary for either network, although it could be useful to tune and debug networks by examining intermediate inferences.

6. Test Results

The following results shows runs of the markov approximation on simple networks with different evidence/query combinations and increasing numbers of iterations. In each section, the error for the given variables are shown, separated by "stepsize" iterations.

```
Markov Approx Test: no evidence

STEP = 20000, MAX = 100000


      a     :   0.1550% ( 19.8450% vs  20.0000%)
      b     :   0.2500% ( 32.0250% vs  32.0000%)
      c     :   0.5499% (  7.9450% vs   8.0000%)
      d     :   0.1350% ( 31.8650% vs  32.0000%)
      e     :   0.3700% ( 61.9700% vs  61.6000%)
      total :   0.1480%

      a     :   0.2675% ( 19.7325% vs  20.0000%)
      b     :   0.3350% ( 31.6650% vs  32.0000%)
      c     :   0.1799% (  8.1800% vs   8.0000%)
      d     :   0.2174% ( 31.7825% vs  32.0000%)
      e     :   0.2674% ( 61.3325% vs  61.6000%)
      total :   0.2535%

      a     :   0.1016% ( 20.1017% vs  20.0000%)
      b     :   0.1450% ( 31.8550% vs  32.0000%)
      c     :   0.1600% (  8.1600% vs   8.0000%)
      d     :   0.1500% ( 32.0150% vs  32.0000%)
      e     :   0.2283% ( 61.3717% vs  61.6000%)
      total :   0.1299%

      a     :   0.3599% ( 20.3600% vs  20.0000%)
      b     :   0.6450% ( 32.6450% vs  32.0000%)
      c     :   0.2649% (  8.2650% vs   8.0000%)
      d     :   0.7587% ( 32.7588% vs  32.0000%)
      e     :   0.3075% ( 61.9075% vs  61.6000%)
      total :   0.4672%

      a     :   0.2400% ( 19.7600% vs  20.0000%)
      b     :   0.4910% ( 31.5090% vs  32.0000%)
      c     :   0.5600% (  7.9440% vs   8.0000%)
      d     :   0.3470% ( 31.6530% vs  32.0000%)
      e     :   0.6500% ( 61.6650% vs  61.6000%)
      total :   0.2398%

Markov Approx Test: prior evidence (A=T)
STEP = 20000, MAX = 100000

      d     :   0.2149% ( 68.2150% vs  68.0000%)
      e     :   0.2650% ( 64.2650% vs  64.0000%)
      total :   0.2400%

      d     :   0.1099% ( 68.1100% vs  68.0000%)
      e     :   0.1924% ( 63.8075% vs  64.0000%)
      total :   0.1512%

      d     :   0.2016% ( 67.7983% vs  68.0000%)
      e     :   0.1950% ( 64.1950% vs  64.0000%)
      total :   0.1983%

      d     :   0.3025% ( 67.6975% vs  68.0000%)
      e     :   0.2024% ( 63.7975% vs  64.0000%)
      total :   0.2524%
```

```
      d     :   0.9100% ( 67.9090% vs  68.0000%)
      e     :   0.1780% ( 64.1780% vs  64.0000%)
      total :   0.1345%

Markov Approx Test: prior evidence (A=F)
STEP = 20000, MAX = 100000

      d     :   0.3500% ( 23.0350% vs  23.0000%)
      e     :   0.1749% ( 60.8250% vs  61.0000%)
      total :   0.1049%

      d     :   0.1249% ( 23.1250% vs  23.0000%)
      e     :   0.9749% ( 60.9025% vs  61.0000%)
      total :   0.1112%

      d     :   0.3450% ( 23.3450% vs  23.0000%)
      e     :   0.6833% ( 61.0683% vs  61.0000%)
      total :   0.2066%

      d     :   0.2524% ( 23.2525% vs  23.0000%)
      e     :   0.1487% ( 60.8513% vs  61.0000%)
      total :   0.2006%

      d     :   0.4370% ( 22.5630% vs  23.0000%)
      e     :   0.4000% ( 61.0400% vs  61.0000%)
      total :   0.2385%

Markov Approx Test: posterior evidence (D=T, E=T)
STEP = 20000, MAX = 100000

      a     :   0.2537% ( 42.7150% vs  42.9688%)
      total :   0.2537%

      a     :   0.1512% ( 43.1200% vs  42.9688%)
      total :   0.1512%

      a     :   0.3437% ( 42.6250% vs  42.9688%)
      total :   0.3437%

      a     :   0.1437% ( 42.8250% vs  42.9688%)
      total :   0.1437%

      a     :   0.1167% ( 42.8520% vs  42.9688%)
      total :   0.1167%

Markov Approx Test: posterior evidence (D=T, E=F)
STEP = 20000, MAX = 100000

      a     :   0.6283% ( 42.2950% vs  41.6667%)
      total :   0.6283%

      a     :   0.2791% ( 41.3875% vs  41.6667%)
      total :   0.2791%

      a     :   0.2283% ( 41.8950% vs  41.6667%)
      total :   0.2283%

      a     :   0.2295% ( 41.8963% vs  41.6667%)
      total :   0.2295%
```

```
      a     :   0.1066% ( 41.6560% vs  41.6667%)
      total :   0.1066%

Markov Approx Test: posterior evidence (D=F, E=T)
STEP = 20000, MAX = 100000

      a     :   0.2523% (  9.9800% vs   9.7276%)
      total :   0.2523%

      a     :   0.1098% (  9.8375% vs   9.7276%)
      total :   0.1098%

      a     :   0.6095% (  9.6667% vs   9.7276%)
      total :   0.6095%

      a     :   0.1673% (  9.8950% vs   9.7276%)
      total :   0.1673%

      a     :   0.3337% (  9.7610% vs   9.7276%)
      total :   0.3337%

Markov Approx Test: posterior evidence (D=F, E=F)
STEP = 20000, MAX = 100000

      a     :   0.2064% (  9.1350% vs   8.9286%)
      total :   0.2064%

      a     :   0.3089% (  9.2375% vs   8.9286%)
      total :   0.3089%

      a     :   0.2919% (  8.6367% vs   8.9286%)
      total :   0.2919%

      a     :   0.1235% (  8.8050% vs   8.9286%)
      total :   0.1235%

      a     :   0.1257% (  8.9160% vs   8.9286%)
      total :   0.1257%

Markov Approx Test: mid evidence (B=T, C=T)
STEP = 20000, MAX = 100000

      a     :   0.1149% ( 80.1150% vs  80.0000%)
      d     :   0.6500% ( 79.9350% vs  80.0000%)
      e     :   0.4249% ( 80.4250% vs  80.0000%)
      total :   0.2016%

      a     :   0.2299% ( 80.2300% vs  80.0000%)
      d     :   0.2250% ( 79.9775% vs  80.0000%)
      e     :   0.3999% ( 80.0400% vs  80.0000%)
      total :   0.9749%

      a     :   0.2666% ( 79.7333% vs  80.0000%)
      d     :   0.1949% ( 79.8050% vs  80.0000%)
      e     :   0.6500% ( 79.9350% vs  80.0000%)
      total :   0.1755%

      a     :   0.9500% ( 80.0950% vs  80.0000%)
```

```
      d     :   0.4750% ( 80.0475% vs  80.0000%)
      e     :   0.2000% ( 80.2000% vs  80.0000%)
      total :   0.1141%

      a     :   0.6099% ( 80.0610% vs  80.0000%)
      d     :   0.4400% ( 80.0440% vs  80.0000%)
      e     :   0.1580% ( 79.8420% vs  80.0000%)
      total :   0.8766%

Markov Approx Test: mid evidence (B=T, C=F)
STEP = 20000, MAX = 100000

      a     :   0.2457% ( 45.9600% vs  45.7143%)
      d     :   0.5600% ( 79.4400% vs  80.0000%)
      e     :   0.3649% ( 59.6350% vs  60.0000%)
      total :   0.3902%

      a     :   0.4757% ( 46.1900% vs  45.7143%)
      d     :   0.2475% ( 80.2475% vs  80.0000%)
      e     :   0.1175% ( 60.1175% vs  60.0000%)
      total :   0.2802%

      a     :   0.1676% ( 45.5467% vs  45.7143%)
      d     :   0.3833% ( 79.9617% vs  80.0000%)
      e     :   0.1516% ( 59.8483% vs  60.0000%)
      total :   0.1192%

      a     :   0.2119% ( 45.9263% vs  45.7143%)
      d     :   0.8999% ( 80.0900% vs  80.0000%)
      e     :   0.2712% ( 59.7288% vs  60.0000%)
      total :   0.1910%

      a     :   0.2092% ( 45.5050% vs  45.7143%)
      d     :   0.1159% ( 80.1160% vs  80.0000%)
      e     :   0.2519% ( 59.7480% vs  60.0000%)
      total :   0.1924%

Markov Approx Test: mid evidence (B=F, C=T)
STEP = 20000, MAX = 100000

      a     :   0.3500% ( 20.3500% vs  20.0000%)
      d     :   0.3250% ( 79.6750% vs  80.0000%)
      e     :   0.3000% ( 80.3000% vs  80.0000%)
      total :   0.3250%

      a     :   0.8249% ( 19.9175% vs  20.0000%)
      d     :   0.3649% ( 80.3650% vs  80.0000%)
      e     :   0.3474% ( 79.6525% vs  80.0000%)
      total :   0.2649%

      a     :   0.7999% ( 19.9200% vs  20.0000%)
      d     :   0.7166% ( 79.9283% vs  80.0000%)
      e     :   0.2049% ( 79.7950% vs  80.0000%)
      total :   0.1188%

      a     :   0.1450% ( 20.1450% vs  20.0000%)
      d     :   0.2312% ( 80.2313% vs  80.0000%)
      e     :   0.8499% ( 80.0850% vs  80.0000%)
      total :   0.1537%
```

```
      a     :   0.5099% ( 20.0510% vs  20.0000%)
      d     :   0.7100% ( 80.0710% vs  80.0000%)
      e     :   0.1899% ( 79.8100% vs  80.0000%)
      total :   0.1039%

Markov Approx Test: mid evidence (B=F, C=F)
STEP = 20000, MAX = 100000

      a     :   0.3499% (  5.0350% vs   5.0000%)
      d     :   0.4999% (  5.0500% vs   5.0000%)
      e     :   0.2749% ( 59.7250% vs  60.0000%)
      total :   0.1199%

      a     :   0.1500% (  4.9850% vs   5.0000%)
      d     :   0.2249% (  5.0225% vs   5.0000%)
      e     :   0.2000% ( 60.2000% vs  60.0000%)
      total :   0.7916%

      a     :   0.2333% (  5.0233% vs   5.0000%)
      d     :   0.1100% (  4.8900% vs   5.0000%)
      e     :   0.8666% ( 59.9133% vs  60.0000%)
      total :   0.7333%

      a     :   0.2124% (  5.0213% vs   5.0000%)
      d     :   0.5000% (  4.9950% vs   5.0000%)
      e     :   0.6499% ( 59.9350% vs  60.0000%)
      total :   0.3041%

      a     :   0.4700% (  4.9530% vs   5.0000%)
      d     :   0.1229% (  5.1230% vs   5.0000%)
      e     :   0.2000% ( 60.0200% vs  60.0000%)
      total :   0.6333%

Markov Approx Test: both evidence (A=T, D=T, E=T)
STEP = 20000, MAX = 100000

      b     :   0.1240% ( 93.2150% vs  93.0909%)
      c     :   0.3090% ( 29.0600% vs  29.0909%)
      total :   0.7750%

      b     :   0.5909% ( 93.0850% vs  93.0909%)
      c     :   0.2659% ( 29.1175% vs  29.0909%)
      total :   0.1624%

      b     :   0.2075% ( 93.1117% vs  93.0909%)
      c     :   0.6742% ( 29.1583% vs  29.0909%)
      total :   0.4409%

      b     :   0.5909% ( 93.1500% vs  93.0909%)
      c     :   0.8284% ( 29.1738% vs  29.0909%)
      total :   0.7096%

      b     :   0.5990% ( 93.0310% vs  93.0909%)
      c     :   0.1790% ( 29.2700% vs  29.0909%)
      total :   0.1194%

Markov Approx Test: both evidence (A=T, D=T, E=F)
STEP = 20000, MAX = 100000
```

```
      b     :   0.1699% ( 96.1700% vs  96.0000%)
      c     :   0.1933% ( 13.1400% vs  13.3333%)
      total :   0.1816%

      b     :   0.2999% ( 96.0300% vs  96.0000%)
      c     :   0.2208% ( 13.1125% vs  13.3333%)
      total :   0.1254%

      b     :   0.1650% ( 95.8350% vs  96.0000%)
      c     :   0.9666% ( 13.4300% vs  13.3333%)
      total :   0.1308%

      b     :   0.3375% ( 95.9663% vs  96.0000%)
      c     :   0.1595% ( 13.1738% vs  13.3333%)
      total :   0.9666%

      b     :   0.2999% ( 96.0300% vs  96.0000%)
      c     :   0.3633% ( 13.2970% vs  13.3333%)
      total :   0.3316%

Markov Approx Test: both evidence (A=T, D=F, E=T)
STEP = 20000, MAX = 100000

      b     :   0.1200% ( 51.0800% vs  51.2000%)
      c     :   0.6999% ( 16.0700% vs  16.0000%)
      total :   0.9499%

      b     :   0.1074% ( 51.3075% vs  51.2000%)
      c     :   0.1387% ( 16.0000% vs  16.0000%)
      total :   0.5374%

      b     :   0.2166% ( 51.2217% vs  51.2000%)
      c     :   0.2333% ( 15.9767% vs  16.0000%)
      total :   0.2250%

      b     :   0.3499% ( 51.1650% vs  51.2000%)
      c     :   0.9124% ( 16.0913% vs  16.0000%)
      total :   0.6312%

      b     :   0.2299% ( 51.2230% vs  51.2000%)
      c     :   0.1600% ( 15.9840% vs  16.0000%)
      total :   0.1950%

Markov Approx Test: both evidence (A=T, D=F, E=F)
STEP = 20000, MAX = 100000

      b     :   0.3700% ( 47.6300% vs  48.0000%)
      c     :   0.1883% (  6.8550% vs   6.6667%)
      total :   0.2791%

      b     :   0.5499% ( 48.0550% vs  48.0000%)
      c     :   0.8833% (  6.7550% vs   6.6667%)
      total :   0.7166%

      b     :   0.2149% ( 48.2150% vs  48.0000%)
      c     :   0.1233% (  6.7900% vs   6.6667%)
      total :   0.1691%
```

```
      b     :   0.5012% ( 48.5013% vs  48.0000%)
      c     :   0.1183% (  6.7850% vs   6.6667%)
      total :   0.3097%

      b     :   0.1480% ( 48.1480% vs  48.0000%)
      c     :   0.7166% (  6.5950% vs   6.6667%)
      total :   0.1098%

Markov Approx Test: both evidence (A=F, D=T, E=T)
STEP = 20000, MAX = 100000

      b     :   0.2756% ( 67.1250% vs  66.8493%)
      c     :   0.1271% ( 22.0450% vs  21.9178%)
      total :   0.2014%

      b     :   0.1681% ( 66.8325% vs  66.8493%)
      c     :   0.7469% ( 21.9925% vs  21.9178%)
      total :   0.4575%

      b     :   0.1743% ( 66.6750% vs  66.8493%)
      c     :   0.1421% ( 22.0600% vs  21.9178%)
      total :   0.1582%

      b     :   0.7068% ( 66.9200% vs  66.8493%)
      c     :   0.9780% ( 21.8200% vs  21.9178%)
      total :   0.8424%

      b     :   0.1576% ( 67.0070% vs  66.8493%)
      c     :   0.4519% ( 21.9630% vs  21.9178%)
      total :   0.1014%

Markov Approx Test: both evidence (A=F, D=T, E=F)
STEP = 20000, MAX = 100000

      b     :   0.1842% ( 74.4700% vs  74.2857%)
      c     :   0.7380% (  9.4500% vs   9.5238%)
      total :   0.1290%

      b     :   0.5714% ( 74.2800% vs  74.2857%)
      c     :   0.8130% (  9.4425% vs   9.5238%)
      total :   0.4351%

      b     :   0.1107% ( 74.1750% vs  74.2857%)
      c     :   0.7285% (  9.5967% vs   9.5238%)
      total :   0.9178%

      b     :   0.7571% ( 74.2100% vs  74.2857%)
      c     :   0.7494% (  9.5988% vs   9.5238%)
      total :   0.7532%

      b     :   0.2632% ( 74.5490% vs  74.2857%)
      c     :   0.1438% (  9.3800% vs   9.5238%)
      total :   0.2035%

Markov Approx Test: both evidence (A=F, D=F, E=T)
STEP = 20000, MAX = 100000

      b     :   0.1986% (  5.0600% vs   5.2586%)
      c     :   0.9137% (  1.7150% vs   1.7241%)
```

```
      total :   0.1038%

      b     :   0.6887% (  5.3275% vs   5.2586%)
      c     :   0.8336% (  1.8075% vs   1.7241%)
      total :   0.7612%

      b     :   0.8804% (  5.3467% vs   5.2586%)
      c     :   0.6586% (  1.7900% vs   1.7241%)
      total :   0.7695%

      b     :   0.8612% (  5.1725% vs   5.2586%)
      c     :   0.2586% (  1.7500% vs   1.7241%)
      total :   0.5599%

      b     :   0.8737% (  5.3460% vs   5.2586%)
      c     :   0.2113% (  1.7030% vs   1.7241%)
      total :   0.5425%

Markov Approx Test: both evidence (A=F, D=F, E=F)
STEP = 20000, MAX = 100000

      b     :   0.3803% (  5.0600% vs   5.0980%)
      c     :   0.5359% (  0.6000% vs   0.6535%)
      total :   0.4581%

      b     :   0.1130% (  4.9850% vs   5.0980%)
      c     :   0.1109% (  0.6425% vs   0.6535%)
      total :   0.6206%

      b     :   0.8803% (  5.0100% vs   5.0980%)
      c     :   0.4192% (  0.6116% vs   0.6535%)
      total :   0.6498%

      b     :   0.4821% (  5.1463% vs   5.0980%)
      c     :   0.2859% (  0.6250% vs   0.6535%)
      total :   0.3840%

      b     :   0.6303% (  5.0350% vs   5.0980%)
      c     :   0.4059% (  0.6130% vs   0.6535%)
      total :   0.5181%
```

The following output shows the results of bucket-tree variable elimination on the coma network.

```
a = T

states of 'e' (2):
         T |         F || d
  44.0000% |  24.0000% || T
  20.0000% |  12.0000% || F
d:  68.0000% = T
    32.0000% = F
e:  64.0000% = T
    36.0000% = F

states of 'd' (2):
         T |         F || e
  68.7500% |  31.2500% || T
```

```
  66.6667% |  33.3333% || F
states of 'e' (2):
         T |         F || d
  64.7059% |  35.2941% || T
  62.5000% |  37.5000% || F

a = F

states of 'e' (2):
         T |         F || d
  14.6000% |   8.4000% || T
  46.4000% |  30.6000% || F
d:  23.0000% = T
    77.0000% = F
e:  61.0000% = T
    39.0000% = F

states of 'd' (2):
         T |         F || e
  23.9344% |  76.0656% || T
  21.5385% |  78.4615% || F

states of 'e' (2):
         T |         F || d
  63.4783% |  36.5217% || T
  60.2597% |  39.7403% || F
```

Results on the accuracy of the poker networks in practice can be demonstrated, but not written in text form at the present time. In short, the pre-flop network has been tested in multiple situations and responds in perfect accordance with poker experts' advice. The post-flop network accurately predicts draw probabilities based on outs, and chooses the correct action given pot odds. Active tests on its accuracy in predicting the hole cards held by opponents can only be evaluated by the net winnings, because in on line poker only the winner's hand is shown.

The final test was to be to integrate the *Gambler* with a body of code I have previously written which is able to seamlessly integrate with on line poker rooms and to be able to play many tables simultaneously, without human supervision. The initial (monetary) return from *Gambler* has been amazing.

7. Conclusions and Future Work

Accuracy in the Bayesian network, so far as it can be measured, has far surpassed my initial expectations of its ability to choose the correct answer without much information. Its ability to mimic the advice of experts is the equivalent of what would be called great intuition in a human player. The advantage of a program which can be intuitive is that programs do not tire or second-guess themselves. The post-flop network contains many distributions which could be tuned to produce slightly different responses given position and odds, but the current tuning is producing net-winning strategy.

For further work, it would be helpful to be able to investigate the interior variables of the post-flop network to determine if any gross adjustments could be made to its strategy. For instance, adding early-/mid-/late- action variables might improve the strategy by adding an intuitive element similar to the pre-flop network. In addition, when the AI loses games, the winner's hand is shown. Research should be done into automatic belief network learning to integrate the average differences between the real and projected opponent's hand into the distributions of the network, in effect learning how opponents react to their own observed variables, and in turn being better able to predict what cards they hold. This technique can be applied globally, to tune the network overall, or locally, to tune it temorarily for poker rooms, poker tables, tournaments, or even frequently recurring individual opponents to provide a sort of personal “profile” on players.

In addition, the evaluation method could be improved in various ways. A possible evaluation method for any Bayesian network is to use a Query-DAG, which is an analogous network made up of arithmetic operations which are computed using a normal Bayesian network solution, such as bucket-tree variable elimination. The advantage of this network is that changes in evidence nodes can be propagated through the network without involving many of the other variables. Since the same network is used in the three post-flop rounds, this could greatly reduce the overall computation time, although space guarantees for the QDAG algorithm can be no better the the guarantees provided by the method which generated them, in this case, bucket-trees.

Bibliography

[Charniak, 1991] Eugene Charniak. "Bayesian Networks Without Tears." *AI Magazine (Winter:50-63, pub. American Association for Artificial Intelligence),* 1991.

[Cheng, 2000] Jian Cheng and Marek J. Druzdel. "Latin Hypercube Sampling in Bayesian Networks." *Proceedings of the Thirteenth Florida Artificial Intelligence Research Society Conference (FLAIRS 2000), eds. Jim Etheredge and Bill Manaris (287-292),* 2000.

[Cheng, 2001] Jian Cheng. "Efficient Stochastic Sampling Algorithms for Bayesian Networks." *University of Pittsburgh, Department of Information Science and Telecommunications*, 2001.

[Cozman, 2000] Fabio Gagliardi Cozman. "Generalizing Variable Elimination in Bayesian Networks." *Workshop on Probabilistic Reasoning in Artificial Intelligence, Atibaia, Brazil,* 2000.

[Darwiche, 1997] Adnan Darwiche and Gregory Provan. "Query DAGs: A Practical Paradigm for Implementing Belief-Network Inference." *Journal of Artificial Intelligence Research (6:147-176),* 1997.

[Jones, 2005] Lee Jones. "Winning Low-Limit Hold'em." USA: Conjelco, 2005.

[Sklansky, 1999] David Sklansky. "Hold'em Poker for Advanced Players." (3rd ed.) USA: Two Plus Two Pub., 1999.

[Wang, 2003] Changyun Wang. "Bayesian Belief Network Simulation." *Florida State University, Department of Computer Science,* 2003.