Strategy Report: Team IHTFG

Milo Knowles '19 and Trevor McMichael '19

I. Overall Strategy

Because poker is a game of incomplete information, a good pokerbot must make the most of the information it *does* have. The only absolute information a pokerbot has is the set of cards in front of it. As we developed our strategy, we kept this in mind, and made sure our bot could make the most informed decision possible, given the cards in its hand and the cards on the board.

We believe that the most valuable information a pokerbot can wield is its probability of beating the opponent. In a match between two pokerbots with similar betting strategies, the one that can better determine its probability of winning will almost always win the game. We focused our efforts accordingly, beginning by developing an algorithm that could calculate our probability of winning given any board and any hand. Using this algorithm as a foundation for our bot, we built a betting strategy on top of it.

II. Optimized brute-force probability calculator

At the core of our bot's decision making is an algorithm that calculates our probability of winning any given hand. Using the cards in our hand and on the board, the algorithm determines the best hand we are currently holding (i.e, two-pair of 10s and Ks). Next, it considers every possible combination of two cards in the remaining deck, which are all of the "theoretical hands" for the opponent. For each theoretical opponent hand, our algorithm then evaluates whether that hand could beat ours. We then derive our probability of winning by the following:

$$P_{win} = 100 * \left(100 - \frac{\#\,WINNING\ OPP.HANDS}{\#\,TOTAL\ HANDS}\right)$$

However, because the style of poker in this tournament is Omaha Poker, this win probability is not entirely accurate. In fact, it always gives an overestimate of our own probability of winning.

Why? Because the opponent has *four* cards in their hand, not two. Although they may only use two cards from their hand, they can *choose* any two cards from a hand of four. The fact that they can make this choice significantly changes how probabilities must be calculated.

To prove why this is so, consider the following. Say that the flop has just occurred. For the three cards that are on the board, there exist two cards in the deck which are the *best* two cards for that board. These two best cards can be paired with 990 combinations of two other cards to make a hand of four cards. If you were dealt any of these 990 four-card hands, you would *always* choose the best-two cards to play.

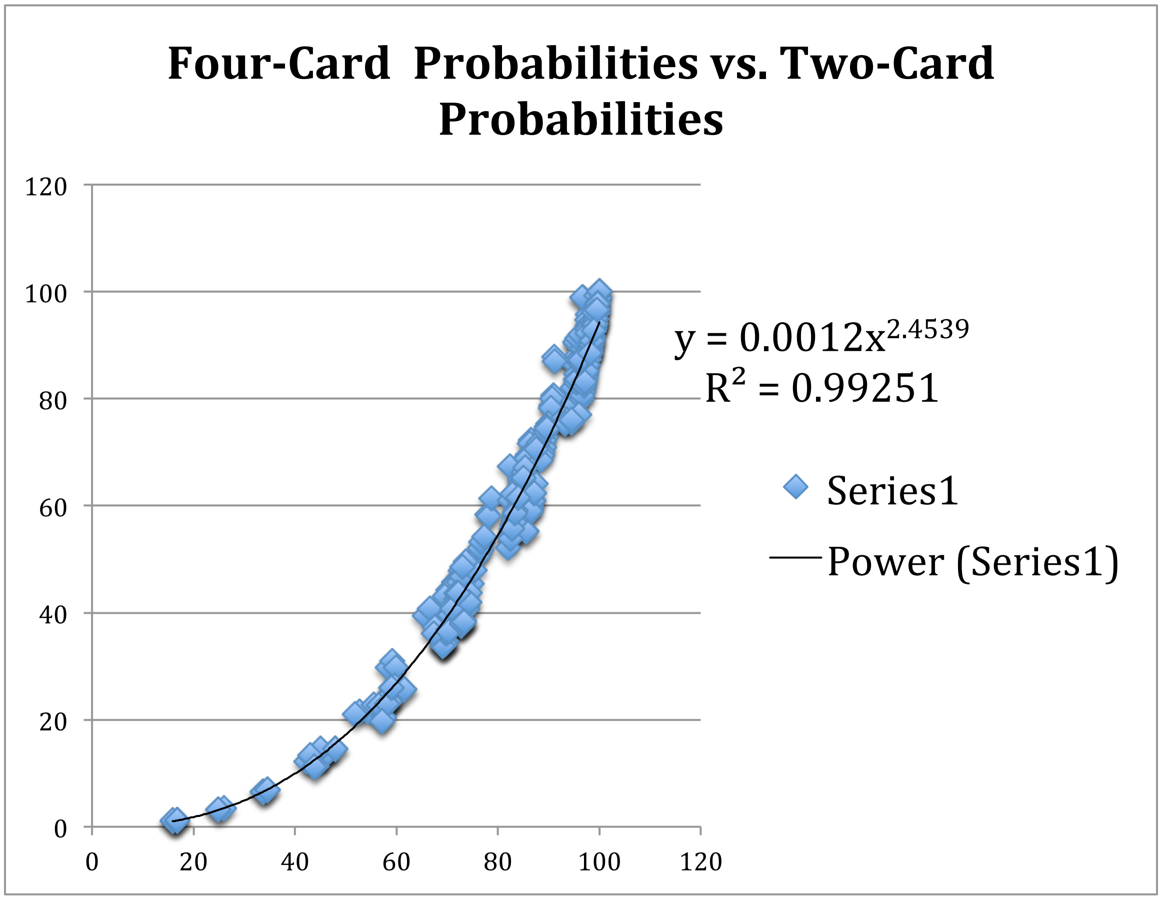
Now consider the *second-best* two cards in the deck. Out of the 990 four-card hands that contain the second-best two cards, you would choose the second-best two cards 889 times. The only scenario in which you would not choose the second-best two cards is when you also happen to hold the best-two cards in the same hand.

This pattern continues: the third-best two-card combination would be chosen 888/990 times, and the fourth 887/990 times. For the 990th best two card combination, which is the *worst* in the deck, you would choose these two cards 0/990 times, because you would always have two other cards in your hand that are better. Essentially, this means that the opponent can never have the worst hand in the deck. Interestingly, this also leads to the conclusion that the opponent is several hundred times more likely to end up with the best hand in the deck than the worst.

The conclusion that can be made is that better hands occur *more frequently* for the opponent; the better the hand, the more outcomes there are in which the opponent chooses the hand. Better hands have a higher associated weight, and lower hands have a lower associated weight.

After reaching this conclusion, we realized that our brute force algorithm, which evaluates every *two-card* theoretical opponent hand for the opponent, is inaccurate. To rectify this problem, we had two options: (1) modify our algorithm to evaluate every *four-card* theoretical opponent hand, or (2) find a way to correlate two-card probabilities to more accurate four-card probabilities. We chose the latter. After the flop, there are 148,995 four-card combinations of four cards. From a computational standpoint, this would make our algorithm impossible to implement given the time constraint of 0.2 seconds per hand.

Luckily, there *is* a correlation between probabilities calculated using two-card hands and probabilities calculated using four-card hands. We simulated 300 hands using a two-card algorithm and a four-card algorithm. Plotting this data reveals a clear relationship between two-card probabilities and four-card probabilities.



Clearly, there is a predictable relationship between probabilities found through considering two-card combinations and four-card combinations. Therefore, any four-card probability, which is the *exact* probability, can be accurately estimated using a two-card probability using the following the equation:

$$P_{exact} \approx 0.0012(P_{2card})^{2.4539}$$

Using this equation, our algorithm can evaluate our probability of victory quickly *and* accurately. The algorithm only needs to iterate over 990 two-card combinations, rather than 148,995 four-card combinations. We believe that our algorithm yields a probability of victory with enough certainty to base the majority of our decision-making around.

III. Betting Strategy

Preflop: Because there is very little information with which to make a decision, our preflop strategy is very simple. We assign points to our hand based on its favorability. Highcards (i.e A,K,Q) are worth 1, 0.5, and 0.25 points, respectively. Pairs are worth 3 points. If our hand is double-suited, which is highly favorable in Omaha, we award our hand 5 points. If our hand is four-suited, which is highly unfavorable in Omaha, we subtract one point from our hand.

If our hand value is above 5 points, we raise by a small amount. If our hand between 0 and 5 we check or call. If our hand is below zero, we check or fold.

Postflop: Admittedly, our betting strategy for the postflop is somewhat arbitrary in comparison to our brute-force probability calculator. Our postflop betting strategy relies heavily on two parameters: our probability of victory and our "look ahead" probability. The probability of victory is determined by the cards we are currently holding. The "look ahead" probability is the likelihood that our hand will improve to a straight or better on the *next* card.

If our probability of victory is above 90%, we raise or by 40% of the pot on the flop and turn, and 75% of the pot on the river. Our raise percentage is much higher on the river because there is a large amount of certainty in our win probability at that point. If our probability of victory is between 80% and 90%, we raise or bet by 25% of the pot on the flop and turn, and 50% of the pot on the river.

Below a 60% probability of victory, we call or check. Below a 30% probability, we checkfold.

Our "look ahead" probability, or the probability that we improve to a straight or better on the next card, helps make our betting strategy more intelligent. When there is greater than a 20% "look ahead", we bet/raise moderately, to try to maximize the reward we could achieve from improving to a much better hand.

There are also limits set within our entire betting strategy. Depending on our bots probability of victory, it will only call within a certain amount. If our hand does not have a high likelihood of victory, but the opponent has checked several times in a row, we bet more aggressively to try to push the opponent out of the pot.

IV. Opportunities for Further Improvement

We believe that the largest opportunities for further improvement are in our betting strategy. Right now, because we had very little time to test our bot against other bots, our betting strategy used somewhat arbitrary thresholds for raising, calling, checking, and folding. Although our brute-force probability of calculator could determine our probability of victory to a high degree of accuracy, we have not yet implemented a decision-making strategy that uses this probability of victory to its full extent. Extensive testing and analysis will be necessary to fine-tune our betting strategy.

In addition, our bot could benefit greatly from an exploitive strategy. Right now, it does not "learn" as it plays. Given adequate time, we would like to improve our bot's strategy so that it adapts its play-style based on its opponent. During early testing of our bot, we noticed that it had many critical flaws. For instance, it often engaged in “raise wars” with opponents who were very aggressive. An adaptive bot could notice the opponent’s aggressiveness and make appropriate adjustments to its own betting strategy.