Texas Hold'em

Tobias Dreher, Dmitriy Zharkov

September 2012

Department of Computer and Information Science
Faculty of Information Technology, Mathematics and Electrical Engineering
Norwegian University of Science and Technology

Contents

1 Introduction

There are a lot different variations of poker games. The most popular is Texas Hold'em. Due to its imperfect information and stochastic game play it is difficult to create a reliable artificial intelligence that could compete against human players.

1.1 Aims

The aim of this project is to design and implement a poker-playing simulator for Texas Hold'em. The project is subdivided in three phases.

Phase I Design and implementation of a basic poker engine for k-player Texas Hold'em. Implementation of a simple player who only uses his current card power to choose his action.

Phase II Implementation of a player who acts on the basis of his hand strength. In order to determine a hand's strength a rollout simulator is used.

Phase III The player uses the previous history of his opponents to react on their actions. Therefore an opponent modeler should be implemented. It determines which action an opponent makes and associates it with the game context.

2 Phase I

2.1 Structure of software

In figure 2.1 the basic of the poker engine is shown. The central class is *Game* where each hand is played. Each hand consists of several rounds. Before the first round the hole cards are drawn out of the deck and dealt to each player. After each round of betting the shared cards are drawn out of the deck and committed to *State*.

During each betting round every player is asked to to make a bet (*makeBet()*) which can be *Fold*, *Call* or *Raise*. For reasons of simplicity the action *Check* is considered a *Call* with bet 0. The action made by a player depends on his strategy. In phase I there are two strategies implemented. These are further described in section 2.2.

There are two possibilities how a hand can end. Either only one player is left before the last betting round has finished or there is a showdown after the last betting round. In the first case the only player left gets the hole pot. In the second case the hand powers of all remaining players are compared in the method *findWinners()*. The method uses the class *PowerRanking* to calculate the hand power. The pot is shared among the winners in the method *sharePot()*.

2.2 Strategies

In order to check the correctness of the game engine a *Random Strategy* is implemented. The next step is to implement a more intelligent strategy that makes its decision on the basis of its current card power.

2.2.1 Random Strategy

Random Strategy is the most simple strategy. It has no intelligence at all and chooses its decisions randomly. Firstly it makes a choice between *Fold*, *Call* and *Raise*. In the case of *Raise* the raise amount is calculated randomly between the one-fold and the fourfold of the *Big Blind*.

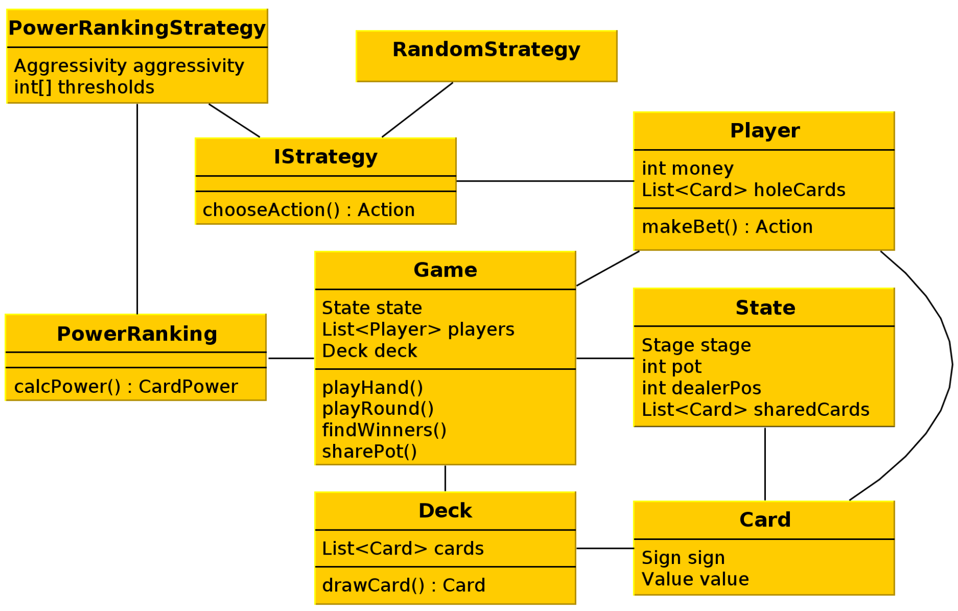


Figure 2.1: Software design of phase I

2.2.2 Power-Ranking Strategy

The action which the *Power-Ranking Strategy* chooses depends on his current card power. If the card power reaches defined thresholds the player is calling or raising. Additionally there are three different levels of aggressivity. The thresholds for each aggressivity level are shown in the table 2.2.2

Aggressivity	Fold	Call	Raise
Conservative	< three of a kind	>= three of a kind	>= straight
Moderate	< two pairs	>= two pairs	>= three of a kind
Risky	< one pair	>= one pair	>= two pairs

Table 2.1: Thresholds for each aggressivity level

Once the decision to raise is made, the amount to raise r is calculated on the basis of the card power p, the aggressivity level a and the size the big blind b with the formula 2.1. The card power is a sequence of numbers where the first number indicates the hand category. $p = 8$ stands for a *Straight Flush* and $p = 0$ stands for *High Card*. The further numbers are needed to break a tie in the same category. The aggressivity level lies in between 1 and 2 where $a = 1$ stands for a conservative player, $a = 2$ for a moderate player and $a = 3$ for a risky player.

$$r = e^{(\frac{p}{3}-3)} * 100 * a + b \tag{2.1}$$

Since there are only 2 cards known during the pre-flop and the hand power can't be calculated the player acts like the player with the random strategy in this stage.

2.3 Results

Table 2.3 shows five 10,000 hand runs where one unintelligent *Random Strategy*-player competes with three *Power-Ranking Strategy*-players.

	Game 1	**Game 2**	**Game 3**	**Game 4**	**Game 5**
Random	-34116	-34456	-32395	-30322	-35032
Power Ranking (risky)	-6298	-122	6552	-2533	-9379
Power Ranking (moderate)	34263	27587	24984	22343	25677
Power Ranking (conservative)	6054	6897	767	10423	18642

Table 2.2: Results of phase I. 10.000 hands on each run.

As expected, the unintelligent random player loses bottomlessly. The player who dominates the game is the moderate *Power-Ranking Strategy*-player. The thresholds at which he folds/call-s/raises seem to be the most reasonable ones.

3 Phase II

3.1 Structure of software

The figure 3.1 shows the additional extensions to the first phase. Here two new strategies are implemented. Both of them use the calculated hand strength to make their decisions. The hand strength calculated by the *Improved Hand Strength Strategy* is more precise than the hand strength of the *Hand Strength Strategy* but needs much more time to calculate. Because the pre flop hand strengths for every card possibility are calculated offline they are saved in files. Therefore a *Writer* and a *Reader* are needed.

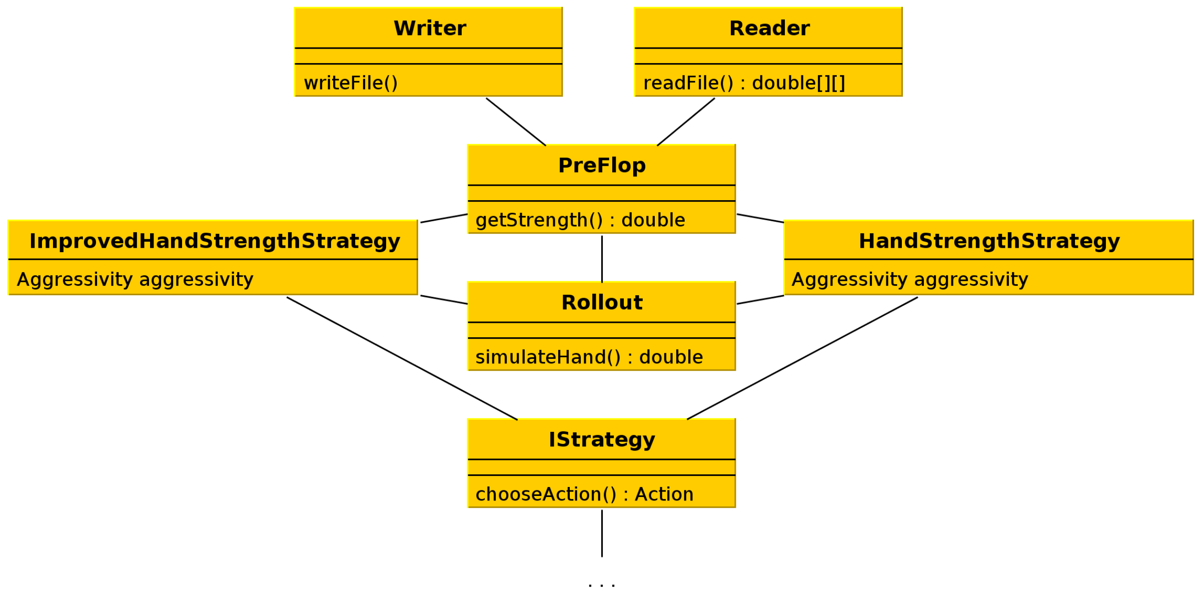


Figure 3.1: Software design of phase II

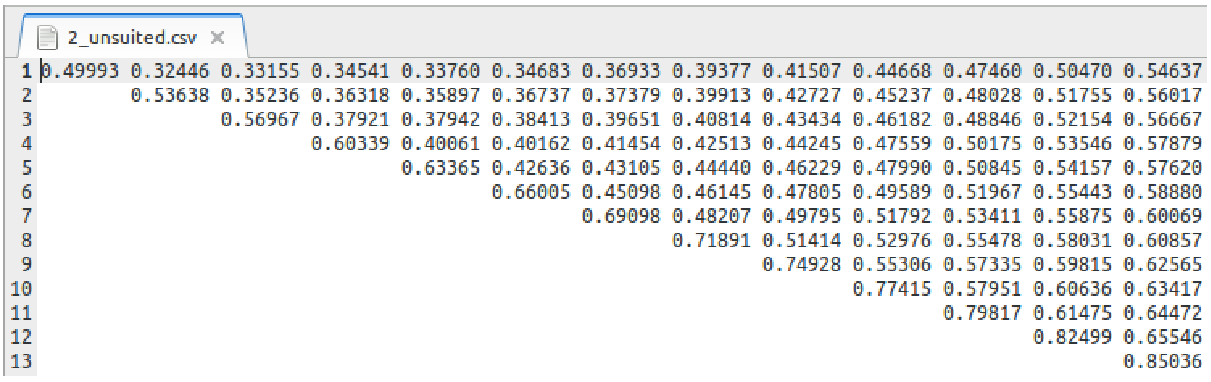
3.2 Rollouts

The calculation of the hand strength is done differently depending whether it's pre flop or not, because at the pre flop there are no shared cards available.

3.2.1 Pre Flop

For the calculation of the hand strength at the pre flop two cards are fixed and the remaining 50 cards are dealt out. This is done many times. The count of the number of times that these two cards win is divided by the number of rollouts to estimate the hand strength of the two cards.

In order to reduce the number of rollout simulations the cards are divided in equivalence classes. For example the strength of a pair of Jacks is independent of the sign. Despite of that there are 169 equivalence classes. For each of these classes 10,000 rollout simulations are performed. This has to be done for every number of players between 2 and 10. So there are over 15 millions simulations to perform. Therefore it is reasonable to do this computation offline and store them in files. The figure 3.2 shows such a file. The value in the top left corner is the hand strength of a pair of twos, the value in the bottom right corner is the hand strength of a pair of aces.



```
0.49993 0.32446 0.33155 0.34541 0.33760 0.34683 0.36933 0.39377 0.41507 0.44668 0.47460 0.50470 0.54637
        0.53638 0.35236 0.36318 0.35897 0.36737 0.37379 0.39913 0.42727 0.45237 0.48028 0.51755 0.56017
                0.56967 0.37921 0.37942 0.38413 0.39651 0.40814 0.43434 0.46182 0.48846 0.52154 0.56667
                        0.60339 0.40061 0.40162 0.41454 0.42513 0.44245 0.47559 0.50175 0.53546 0.57879
                                0.63365 0.42636 0.43105 0.44440 0.46229 0.47990 0.50845 0.54157 0.57620
                                        0.66005 0.45098 0.46145 0.47805 0.49589 0.51967 0.55443 0.58880
                                                0.69098 0.48207 0.49795 0.51792 0.53411 0.55875 0.60069
                                                        0.71891 0.51414 0.52976 0.55478 0.58031 0.60857
                                                                0.74928 0.55306 0.57335 0.59815 0.62565
                                                                        0.77415 0.57951 0.60636 0.63417
                                                                                0.79817 0.61475 0.64472
                                                                                        0.82499 0.65546
                                                                                                0.85036
```

Figure 3.2: Pre Flop rollout for two players and unsuited cards

3.2.2 Flop, Turn and River

After the pre flop stage there are shared cards available. Therefore the card power can be determined. Every possible combination of opposing hole cards is combined with the given shared cards to see if it yields a better hands than the own hand. This calculation is done online.

The problem with this approach is that the missing shared cards at the flop and at the turn aren't considered. Therefore the *Improved Hand Strength Strategy* was implemented. Here the missing shared cards are drawn randomly out of the deck. This is done several times like at the pre flop.

3.3 Strategies

The expectation is that the phase II strategies beat the phase I strategies.

3.3.1 Hand Strength Strategy

The amount of money the player is willing to pay w is calculated on the basis of his hand strength h and the aggressivity level a as shown in formula 3.1. Like in phase I the aggressivity level is a number between 1 and 3. The riskier a player the more is he willing to pay. The maximum amount the a risky player would pay is

$$w = e^{(h*3-3)} * 100 * a + b \tag{3.1}$$

After w is calculated the player's action is determined on the basis of w. If w is smaller than the pot odds multiplied by the amount a call would cost the player folds. That means, that if the player hasn't to pay much (relatively to the pot size) to stay in the game he will pay at least the call. To raise w has to fulfill the following condition **??** where c is the cost of the call and r the number of raises. The more raises there have been, the more likely the player will raise. That's because the player is sure not to frighten away the competitors knowing that they already have raised and don't want to dump their stakes. If there are only few raises the player will more likely call to prevent others from folding.

$$w > \frac{c}{1+r} \tag{3.2}$$

3.3.2 Improved Hand Strength Strategy

The *Improved Hand Strength Strategy* computes the bets the same way like the *Improved Hand Strength Strategy* but it also draws the missing shared cards at the flop and at the turn. Therefore it is significantly slower, but the calculated hand strengths are expected to be more precise.

3.4 Results

Table 3.4 shows five 10,000 hand runs where *Hand Strength Strategy*-players play against each other. The conservative player is the most successful one. The reason for that he is only willing to play a hand if his cards are very good. Therefore he is more likely to win at showdown.

In table 3.4 the difference between the *Hand Strength Strategy* and the *Improved Hand Strength Strategy* is shown. For performance reasons, only 500 hands are played on each run. A slight improvement could be achieved but due to the fact that the number of simulated hands is quite small, this statement is not very reliable.

	Game 1	**Game 2**	**Game 3**	**Game 4**	**Game 5**
Hand Strength (risky)	-248413	-314686	-299379	-256889	-255356
Hand Strength (risky)	-155327	-151467	-154479	-156163	-188908
Hand Strength (moderate)	-29572	49987	91098	10332	41706
Hand Strength (moderate)	84684	48850	51756	63958	60409
Hand Strength (conservative)	200887	173006	149211	222864	198017
Hand Strength (conservative)	147595	194157	161663	115740	143957

Table 3.1: Results of phase II. 10.000 hands on each run (without Improved Hand Strength Players)

	Game 1	**Game 2**	**Game 3**	**Game 4**	**Game 5**
Hand Strength (risky)	-24800	-16712	-5346	-22282	-6416
Hand Strength (moderate)	-4161	4549	-3506	620	2972
Hand Strength (conservative)	5322	12237	3358	3554	6218
Improved Hand Strength (risky)	4285	-5861	797	3077	-9541
Improved Hand Strength (moderate)	6721	10362	190	2490	-170
Improved Hand Strength (conservative)	12628	4517	4498	12532	6921

Table 3.2: Results of phase II. 500 hands on each run.

4 Phase III

4.1 Structure of software

The figure 4.1 shows the implementation details behind the *Opponent Model Strategy*. The *Opponent Logger* tracks all the actions that are made by all players and associates them with the context at which these actions have been made. These actions are stored as a list of *Opponent Entry*-objects.

At the showdown all the players that are still in the game have to show their cards. Now the *Opponent Entry*-list of these players is went through. The player's cards are combined with the share cards and the hand strength is computed. The hand strength is committed to the *Opponent Model* together with the current context and the player's action.

4.2 Model rules

As mentioned above, the *Opponent Model* contains a triple of *Context*, *Player Action* and *Hand Strength* of which the first two are again subdivided.

4.2.1 Context

The *Context* can be considered as the state of the game at a specific point. Features that define a context are

- The stage, whether pre-flop, flop, turn or river
- The number of player that are still in the hand
- The number of raises during the current stage
- The pot odds

However, in order to estimate a player's hand in a given context correctly, the context have to be neither too general nor too specific. Therefore, the mentioned features have to be grouped into classes. For example, the pot odds of 0.20 and 0.21 should be considered the same. The classes that are used for the *Opponent model strategy* are

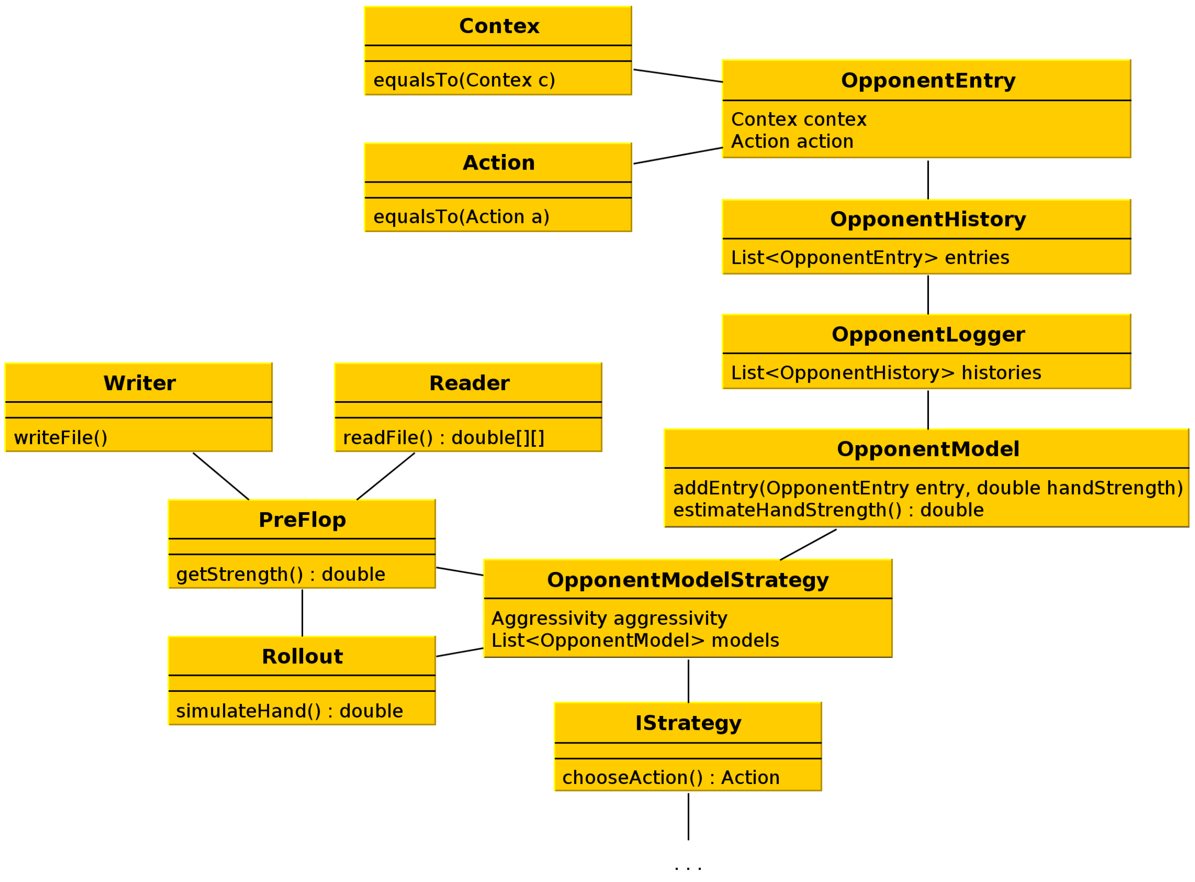


Figure 4.1: Software design of phase III

- **The stage:** pre-flop, flop, turn or river
- **The number of player:** 1 to 3, 4 to 7 and 8 to 10
- **The number of raises:** 1 to 2 and more than 3
- **The pot odds:** 0.0 to 0.3 and more than 0.3

4.2.2 Player Action

The information about a player's action is also subdivided in classes.

- **The action:** call or raise
- **The bet:** 1 to 500 and more than 500

There are no fold entries in the *Opponent Model* because folded cards never reach the showdown and therefore are never shown.

4.3 Opponent Model Strategy

Basically the phase III player acts the same way like phase II player does. At first the amount that a player is willing to pay is calculated the same way. Afterwards the opponents' hand strengths are estimated. Therefore the player retrieves from the model the average hand strength that his opponents had at the same context in the past.

The player compares his own hand strengths with the estimated strengths and gets a number of players who are supposedly better and worse than him. These numbers are used to adjust the player's bet. The new bet is calculated as shown in 4.1, where b' is the new bet, b the old bet, $better$ the number of players who are better and $worse$ the number of player who are worse than the current player.

$$b' = b * (\frac{worse - better}{10}) \tag{4.1}$$

At the beginning, when there are very few hands played and the model has almost no entries, the estimated number of better/worse players will be close to zero. That means that there is no adjustment to the bet and the player plays exactly the way the phase II player does. As the number of played hands rises, the model will have more information about the players and the bet will be adjusted more often.

4.4 Results

As shown in Table 4.4 the *Opponent model strategy*-Player is slightly better than phase II player. The fact that there is not a bigger difference between them may be a hint to not optimally chosen context classes. In general the conservative players are better than the risky ones. The reasons are the same as seen in phase II.

	Game 1	**Game 2**	**Game 3**	**Game 4**	**Game 5**
Hand Strength (risky)	-197017	-202631	-188055	-262959	-196894
Hand Strength (moderate)	-3282	-1537	-13592	-3550	-12102
Hand Strength (conservative)	131998	139472	166900	169885	114067
Opponent Model (risky)	-141772	-192630	-125045	-110525	-153198
Opponent Model (moderate)	54823	112741	33273	71858	99823
Opponent Model (conservative)	155045	144364	126323	135084	148064

Table 4.1: Results of phase III. 10.000 hands on each run.

Table 4.4 shows the competition among all phases. For performance reasons, the *Improved Hand Strength Strategy*-player is not in the competition. The gap between the phase II and phase III player increases, because the phase I player can be easily estimated by the opponent model.

	Game 1	**Game 2**	**Game 3**	**Game 4**	**Game 5**
Random	-659112	-621306	-615979	-611152	-614668
Power Ranking (risky)	-363698	-282221	-264676	-281963	-324449
Power Ranking (moderate)	-156685	-108439	-107078	-80655	-107961
Power Ranking (conservative)	-121214	-129046	-152320	-142944	-149966
Hand Strength (risky)	48976	-22590	-45970	-45916	-15107
Hand Strength (moderate)	22356	111913	155079	145145	133885
Hand Strength (conservative)	255380	205473	271372	189214	238723
Opponent Model (risky)	387196	241850	285574	270680	284309
Opponent Model (moderate)	349258	366101	242860	273993	341882
Opponent Model (conservative)	237244	237975	230835	283306	213029

Table 4.2: Results of phase III. 10.000 hands on each run.

5 Conclusion

The improvements from phase I to phase II are very convincing. That means that the hand strengths is a good basis to make a bet on. Although the gap between phase I and phase II is not as big as expected, an improvement could still be achieved.

One point of criticism is that the bots have never played against human beings. However, the design of the poker engine allows to add a human player without major effort.

The biggest potential of improvements are in the chosen detail settings such as context class thresholds of the phase III player. Alternatively a neural network or an evolutionary algorithm could be used instead of hard coded thresholds.