Speeding-Up Poker Game Abstraction Computation: Average Rank Strength

Luís Filipe Teófilo, Luís Paulo Reis, Henrique Lopes Cardoso
LIACC – Artificial Intelligence and Computer Science Lab., University of Porto, Portugal
Rua Campo Alegre 1021 4169-007 Porto, Portugal
luis.teofilo@fe.up.pt, lpreis@dsi.uminho.pt, hlc@fe.up.pt

Abstract

Some of the most successful Poker agents that participate in the Annual Computer Poker Competition (ACPC) use an almost zero regret strategy: a strategy that approximates a Nash Equilibrium. However, it is still unfeasible to efficiently compute a Nash Equilibrium without some sort of information set abstraction due to the size of Poker's search tree. One popular technique for abstracting Poker information sets is to group hands with similar Expected Hand Strength ($E[HS]$) and thus play them in the same way. For large Poker variants, algorithms like CFR might need to calculate $E[HS]$ billions of times, when the game abstraction is so large that it cannot be pre-computed, implying that $E[HS]$ must be determined online. This way, improving the efficiency of this method would certainly reduce the computation time needed by CFR for these cases. In this paper we describe Average Rank Strength; a technique based on a pre-computed lookup table that speeds up $E[HS]$ computation. Ours results demonstrate speed improvements of about three orders of magnitude and negligible results difference, when compared to the original $E[HS]$.

1. Introduction

For more than a decade and half, the Computer Poker domain has been used as a progress measure for validating extensive-form games research. Several successful techniques have emerged, with special emphasis on case based reasoning and regret minimizing agents. For the latest ones, the Counterfactual Regret Minimization (CFR) [1] and its variations such as CFR-BR [2] are the current state of the art algorithms to find Nash Equilibrium strategies for these type of games.

Despite the CFR breakthrough, it is still unfeasible with the current computational resources to solve very large games like Texas Hold'em Poker (about 3.2×10^{14} information sets in the 2 player Limit version). For that reason, CFR is usually applied on a simplified version of the game through a process called information set abstraction.

Abstraction consists of grouping decision points and act similarly with information sets of the same group. A common method to abstract information sets in Poker is to compute the Expected Hand Strength $E[HS]$ and group hands by that value. Another similar measure is the Expected Hand Strength Squared $E[HS^2]$ which potentiates the hands with higher potential to evolve in future rounds of the game. There are other measures available, but most of them are adaptations or based on $E[HS]$.

In this paper we present a new method to quickly compute the $E[HS]$ – the Average Rank Strength. The new method runs in constant time and is based on lookup tables of pre-computed values of $E[HS]$. This means that the new method is very lightweight in terms of CPU requirements. Moreover, regardless of the need to store the pre-calculated results, the created lookup tables have very low memory requirements, considering today's computers typical RAM size.

The rest of the paper is organized as follows. Section 2 presents the paper's background: definition of hand rank and expected hand strength and how they are computed. Section 3 describes our technique – Average Rank Strength ARS which speeds up the $E[HS]$ computation. Section 4 presents the analysis of our method by indicating the results of speed tests and by comparing the new approach to the original Expected Hand Strength. Finally, conclusions and future perspectives are withdrawn in Section 5.

2. Background

Poker is a popular class of card betting games with similar rules. The most popular and played variant of Poker is currently Texas Hold'em. This variant (or its simplified versions) is also the most used for computer science research since its rules present specific characteristics that allow for new developed approaches to be adapted to other variants with reduced effort [3].

Hand Rank

One important concept in Texas Hold'em rules is the hand and its score. Being Δ the set of all cards in the deck, Φ_i the set of pocket cards of a particular player i and Ω the set of community cards so that $\Phi_1 \cup \Phi_2 \cup \ldots \Phi_n \cup \Omega \subseteq \Delta$, and $\Phi_i \cap \Omega$ for any i is equal to $\emptyset$. Thus, the score function is defined as $s: [\Delta]^5 \rightarrow \mathbb{N}$. For a particular player i, the hand is the union of its pocket cards and the community cards ($\Phi_i \cup \Omega$). Thus, the player's score is given by the rank function, as follows:

$$Rank(i) = \max(\{\forall x \in [\Phi_i \cup \Omega]^5 : s(x)\})$$

There are 9 possible ranks (High Card, One Pair, Two pairs …) and 7462 possible sub-ranks. The relative frequencies of each sub-rank on each Post-Flop round of the game can be seen of Figure 1.

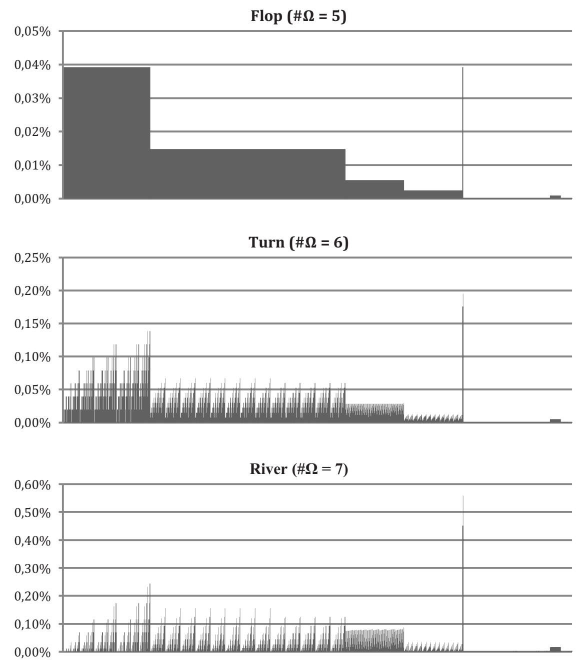

Figure 1. Hand rank relative frequencies in Flop, Turn and River. All possible sub-ranks are represented in the horizontal axis, ordered by their score.

It is possible to observe a stair step layout in the first chart (#Ω = 5). Each stair represents one of the higher level ranks. It is also possible to observe large peeks near the end of each chart. They represent the *Straight* hands, because there are plenty of ways of combining 5, 6 or 7 cards to score a *Straight*, but there are only 10 types of straights (Five high, Six High …).

Programming an algorithm to determine the hand's rank is a trivial task. This can be done using a naïve approach, i.e. using an algorithm that intuitively makes sense and that is humanly readable. However, to compute $E[HS]$, several hand comparisons must be made (see Table 1). Due to the high number of needed hand comparisons, a naïve approach is not recommended.

Table 1. Number of hand comparisons needed to compute E[HS] in each game round, against 1 opponent.

Round	Number of hand comparisons
Pre-Flop	$C(50,5) \times C(45,2) = 2{,}097{,}572{,}400$
Flop	$C(47,5) \times C(45,2) = 1{,}070{,}190$
Turn	$C(46,5) \times C(45,2) = 45{,}540$
River	$C(45,2) = 990$

To improve the speed of hand ranking, pre-computed lookup tables of hand ranks are usually used. There are several known hand evaluators based on lookup tables, but the TwoPlusTwo (TPT) evaluator proved to be the fastest one, experimentally [4]. With Varho enhancement [5], this evaluator results in 7 different lookup tables with 80 MB of total size. With TPT tables, it is possible to rank all possible 5 card hand combinations (2,598,960 hands) in less than 100 ms (in modern CPUs) [6].

TPT represents the cards with integer values. The value of any card is given by:

$$TPTIndex(Rank, Suit) = Rank \times 4 + Suit + 1$$

where Rank is 0 for Two, 1 for Three, ... 12 for Ace, and Suit is 0 for Clubs, 1 for Diamonds, 2 for Hearts and 3 for Spades. To determine the rank of the hand, the lookup tables must be accessed the following way:

$$T_7[T_6[T_5[T_4[T_3[T_2[T_1[53 + C_1] + C_2] + C_3] + C_4] + C_5] + C_6] + C_7]$$

where T_n is the *nth* lookup table and C_n is the *nth* card of the hand. This rank evaluator supports hands with 5, 6 or 7 cards. The order of the hand's cards before performing a lookup is irrelevant.

Expected Hand Strength

The Expected Hand Strength $E[HS]$ is the probability of the current hand of a given player being the best if the game reaches a showdown, against all remaining players. It consists of enumerating all combinations of possible opponents' hands and the remaining hidden board cards and checking if the agent's hand is better than the hands in the enumeration. By counting the number of times the player's hand is better, it is possible to measure the quality of the hand. The *Ahead*, *Tied* and *Behind* functions (defined bellow) determine respectively the number of times the player's hand wins, ties or loses the game:

$$Ahead(i) = \#\{\forall b[\Delta\backslash\Phi_i\backslash\Omega]^{5-\#\Omega}: \forall o \in [\Delta\backslash\Phi_i\backslash\Omega\backslash b]^2: \mathrm{Rank}(\Phi_i + \Omega + b) > \mathrm{Rank}(o + \Omega + b)\}$$

$$Tied(i) = \#\{\forall b[\Delta\backslash\Phi_i\backslash\Omega]^{5-\#\Omega}: \forall o \in [\Delta\backslash\Phi_i\backslash\Omega\backslash b]^2: \mathrm{Rank}(\Phi_i + \Omega + b) = \mathrm{Rank}(o + \Omega + b)\}$$

$$Behind(i) = \#\{\forall b[\Delta\backslash\Phi_i\backslash\Omega]^{5-\#\Omega}: \forall o \in [\Delta\backslash\Phi_i\backslash\Omega\backslash b]^2: \mathrm{Rank}(\Phi_i + \Omega + b) < \mathrm{Rank}(o + \Omega + b)\}$$

The $E[HS]$ for player i against a given number of opponents *n* can be given by:

$$E[HS]_n(i) = \left(\frac{Ahead(i) + \frac{Tied(i)}{2}}{Ahead(i) + Tied(i) + Behind(i)}\right)^n$$

The Expected Hand Strength may be used at any round of the game. However, the number of iterations needed to compute the $E[HS]$ for a single hand at early rounds is very high (see Table 2). Some possible solutions to this problem are:

- Pre-compute $E[HS]$ for all permutations of 2, 5, 6 and 7 cards. Problems: the size of the table would be enormous and incompatible with current available computational resources (see Table 2).
- Pre-compute $E[HS]$ for all combinations of 2, 5, 6 and 7 cards. Problems: the size of the table would still be high (see Table 2) and the hand's cards must be ordered to consult the table.
- Use Monte Carlo sampling by generating a fixed number of possible boards and opponent cards instead of enumerating them all. Problems: the estimation error in the hand evaluation process could send it to another bucket in the abstraction process.
- Pre-compute Monte Carlo sampled $E[HS]$ values. This methodology does not present an advantage over the first two, since the pre-computation despite being very slow, is only performed once.

None of the described methods can generate a table that can be easily stored in RAM memory in current computers (for faster lookups). One possible technique to reduce the tables' overall size is to combine isomorphic hands that vary by a suit rotation. This can shrink the table by approximately an order of magnitude.

Table 2. E[HS] lookup table approximated size considering that each pre-computed value is stored in 8 bytes (double).

Round	Permutations	Combinations
Counting		
Pre-Flop	$P(52,2) = 2652$	$C(52,2) = 1326$
Flop	$P(52,2) \times P(50,3) \approx 3.119 \times 10^8$	$C(52,2) \times C(50,3) \approx 2.599 \times 10^7$
Turn	$P(52,2) \times P(50,3) \approx 1.466 \times 10^{10}$	$C(52,2) \times C(50,4) \approx 3.054 \times 10^8$
River	$P(52,2) \times P(50,3) \approx 6.743 \times 10^{11}$	$C(52,2) \times C(50,3) \approx 2.809 \times 10^9$
Total	$\sim 6.892 \times 10^{11}$	$\sim 3.141 \times 10^9$
Total (shrunken)	$\sim 3.446 \times 10^{10}$	$\sim 1.570 \times 10^8$
Size in GB		
Pre-Flop	~0.00	~0.00
Flop	~2.50	~0.21
Turn	~117.27	~2.44
River	~5394.19	~22.48
Total	~5513.95	~25.13
Total shrunken	~351.38	~3.20

3. Average Rank Strength

In order to improve the efficiency of the $E[HS]$ method, we introduce a new technique called Average Rank Strength (*ARS*). *ARS* consists of using the hand score to estimate the future outcome of the match, without having to generate all card combinations. This is simply done by storing the average $E[HS]$ of a hand per each score in three lookup tables, one for Flop, one for Turn and one for River. Since there are only 7462 possible scores, the lookup table size would be $7462 \times 8 = 59696$ bytes being therefore easily stored in RAM memory for fast $E[HS]$ retrievel.

Storing the average $E[HS]$ values for each rank is not enough; it is crucial to identify the player's pocket cards. To better illustrate this, let us analyze the following hand: A♣ A♦ A♥ K♥ K♠. This hand always scores a Full House despite which two cards belong to the player. However, the hand strength is different for each case (e.g. if player 1 has the two Kings, an player 2 could have the remaining Ace, thus being ahead of player 1. Still, if the player 1 has two Aces, only a Straight Flush would have a higher Hand Strength, and even so only possible in River round).

Introducing a 2nd dimension into the lookup table – the pocket hands id – allows for identifying the player's pocket cards. The pocket hands id is a unique number for a pair of cards, which takes into consideration game's isomorphisms (e.g. A♣ A♦ = A♠ A♥). The total number of possible starting pairs' ids is 167. To quickly obtain the id of a pair the values are stored in a 52×52 pre-computed table named $pairs$. Thus, the id of a given pair can be found in $pairs[Card1][Card2]$.

The total size of the each lookup table is $7462 \times 167 \times 8\ bytes \approx$ 9,97 MB, where 7462 is the number of possible card ranks, 167 the number of unique pairs and 8 the size of a double precision floating point number. The pre computation of a given score is as follows:

$$ARS_{n,r}(C_1, C_2) = \left(\frac{\sum_i X_i \in [\Delta \backslash \Phi\,]^r : Rank(X_i \cup \{C_1, C_2\})}{\#([\Delta \backslash \Phi]^r)}\right)^n$$

where n is the number of opponents, r is the number of community cards and X_i is a distinct subset of size 5 of the deck except the pocket cards. The table lookup process is summarized in Figure 2.

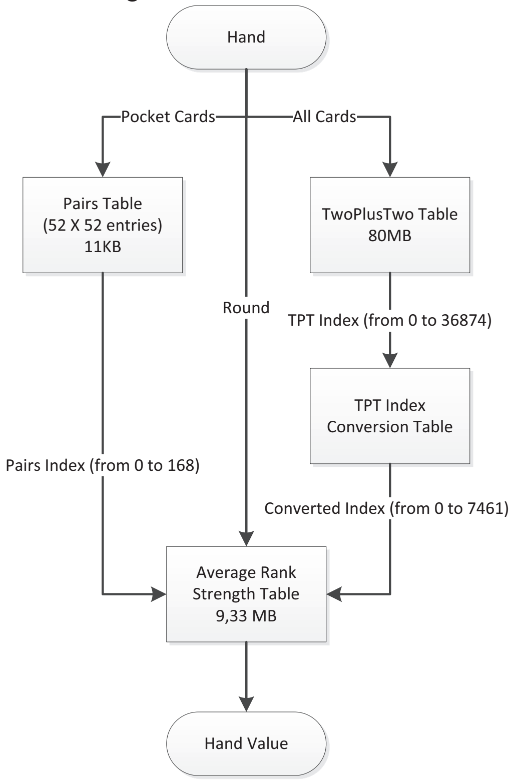
Hand
Pocket Cards
All Cards
Pairs Table (52 X 52 entries) 11KB
TwoPlusTwo Table 80MB
Round
TPT Index (from 0 to 36874)
TPT Index Conversion Table
Pairs Index (from 0 to 168)
Converted Index (from 0 to 7461)
Average Rank Strength Table 9,33 MB
Hand Value

Figure 2. ARS tables hand value lookup process

We used the TwoPlusTwo rank table to compute the index to search in the ARS lookup table (since it is the fastest known rank evaluator). TwoPlusTwo returns an index between 0 and 36874; however, only about 20% of the indexes correspond to a possible rank. We thus created an auxiliary table (similar to the pairs table) so as to convert that index into a number between 0 and 7461, to reduce each lookup table size (from 49.26MB to 9.97MB).

The resulting tables for each round can be seen on Figure 3, in the form of a heat map. The vertical axis represents the unique pair id (ordered), the horizontal axis represents the converted TPT index and the color intensity represents the *ARS* value.

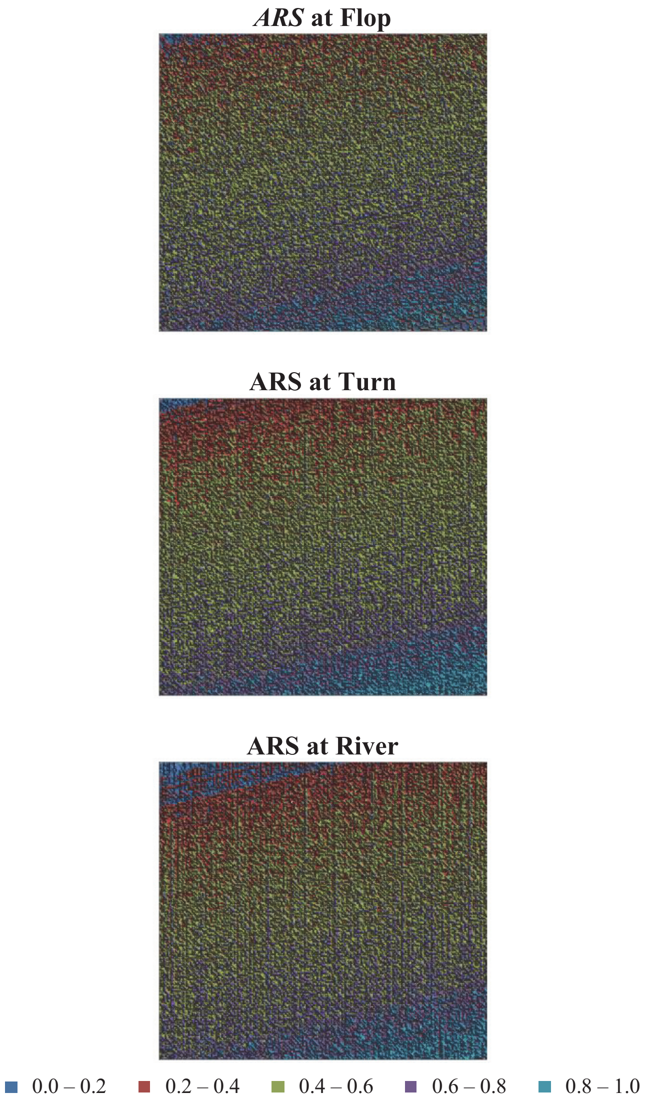

Figure 3. ARS values for Flop, Turn and River

4. Tests and Results

To validate our approach we performed some comparative benchmark tests. *ARS* and *E[HS]* showed no significant result difference even though the *ARS* computed much faster.

Benchmark tests

In order to determine the speed-up factor of the new method against the $E[HS]$ method (with Monte Carlo sampling, 1000 samples), a benchmark test was performed. The test consisted of ranking three 1,000,000 hands pre-computed sequences, one with 5 cards (Flop), one with 6 cards (Turn) and one with 7 cards (River). The tests were performed 1000 times each on an Intel I7-3940XM CPU (4 physical cores) and are presented on Table 3. The obtained standard deviations from the mean are negligible in all cases.

Table 3. Benchmark ARS against E[HS]

Hand rank program	**Round**	**Average elapsed time for 1000 trials in seconds**	
		Non parallel	Parallel (8 cores)
Expected Hand Strength (E[HS])	Flop	387.71	108.90
	Turn	309.18	90.19
	River	263.79	75.98
Average Rank Strength (ARS)	Flop	0.32	0.06
	Turn	0.41	0.09
	River	0.43	0.10
Speedup factor	Flop	1211.59	1815.00
	Turn	754.10	1002.11
	River	613.47	759.8

Our benchmark test demonstrates very promising results, with an average speed-up of 1026.01. Poker agent strategies based on Nash Equilibrium approximation will certainly benefit from this speed improvement because algorithms such as Counterfactual regret minimization need to perform these calculations billions of times (depending on the number of running iterations and the abstraction size). This speed improvement is only useful for CFR if the abstraction of the information sets is done online, instead of being pre-computed. If the game abstraction is pre-computed, the *E*[*HS*] or *ARS* values will not be used directly by CFR. In this case, the use of *ARS* lookup tables would only reduce the time need to compute the abstraction table. This speed-up factor is also useful for agents with other types of strategies (naïve approaches, cased based reasoning, etc…).

Comparison with *E*[*HS*]

We also analyzed the difference between this method and the expected hand strength method. We demonstrate the difference through heat maps where each the axis represent the pocket cards and the color intensity is the average obtained value. The top-right side of the map represents suited card pairs and the bottom-left represents unsuited card pairs. The obtained heat maps for Expected Hand Strength and Average Rank Strength on Pre-Flop are respectively presented on Figures 4 and 5.

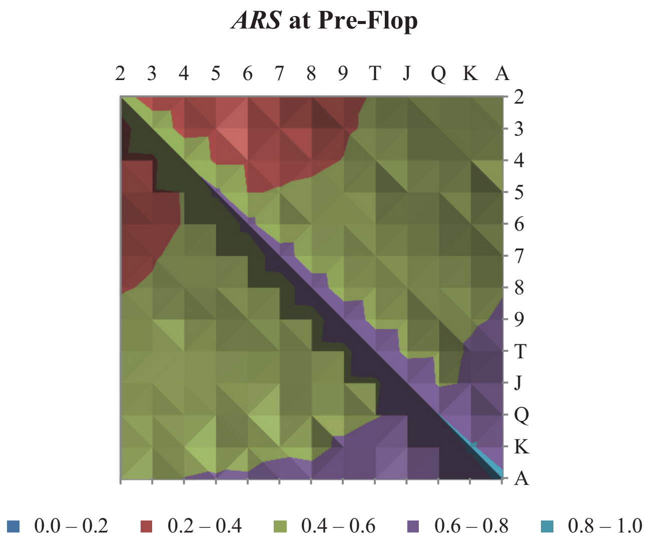

Figure 4. ARS heat map at Pre-Flop

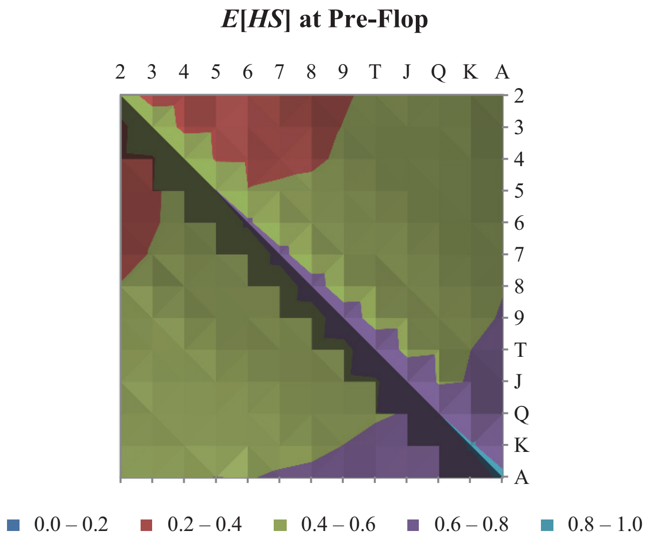

Figure 5. E[HS] heat map at Pre-Flop

This approach not only provides a much faster response to queries – about three orders of magnitude faster – but also it does so with negligible error, as can be seen from the heat maps since the *ARS* charts are very similar to the *E[HS]* ones. The average absolute difference between the two methods is 0.011, the maximum difference found was 0.062 and the summed squared error is 0.039.

5. Conclusions

A new method with a much lower computation time was introduced – Average Rank Strength – which computes similar results to the Expected Hand Strength approach in much less time. *ARS* lookup tables are easy to generate and need relatively low computational requirements, both in memory (about 110MB taking the TwoPlusTwo tables into account) and CPU (the new method is three orders of magnitude faster than the Monte Carlo of *E[HS]*). We believe that future integration of Average Rank Strength with regret minimizing algorithms (when not using game abstraction pre-computation) will contribute towards much lighter Nash Equilibrium strategy computation.

Pre-computed Average Rank Strength lookup tables are available to download at:
http://paginas.fe.up.pt/~pro10020/poker/ars.zip

Acknowledgents. This work was financially supported by FCT – Fundação para a Ciência e a Tecnologia through the Ph.D. Scholarship with reference SFRH/BD/71598/2010.

References

[1] M. Zinkevich, M. Bowling, and N. Burch, “A new algorithm for generating equilibria in massive zero-sum games,” in *Proceedings of the Twenty-Second Conference on Artificial Intelligence (AAAI)*, 2007, pp. 788–793.

[2] M. Johanson, N. Bard, N. Burch, and M. Bowling, “Finding Optimal Abstract Strategies in Extensive-Form Games,” in *Proceedings of the Twenty-Sixth Conference on Artificial Intelligence (AAAI-12)*, 2012, pp. 1371–1379.

[3] D. Billings, A. Davidson, J. Schaeffer, and D. Szafron, “The challenge of poker,” *Artificial Intelligence*, vol. 134, no. 1–2, pp. 201–240, 2002.

[4] L. F. Teófilo, R. Rossetti, L. P. Reis, and H. L. Cardoso, “Simulation and Performance Assessment of Poker Agents,” in *Springer LNCS 7838 (MABS 2012)*, 2013, pp. 69–84.

[5] J. Varho, “7 Card Poker Hand Evaluation,” 2009. [Online]. Available: http://jan.varho.org/?p=99.

[6] L. F. Teófilo, L. P. Reis, and H. L. Cardoso, “Computing Card Probabilities in Texas Hold’em,” in *CISTI’2013 - 8^{a} Conferência Ibérica de Sistemas e Tecnologias de Informação (to appear)*, 2013.