A Near-Optimal Strategy for a Heads-Up No-Limit Texas Hold'em Poker Tournament

Peter Bro Miltersen
University of Aarhus
Åbogade 34, Århus, Denmark
bromille@daimi.au.dk

Troels Bjerre Sørensen
University of Aarhus
Åbogade 34, Århus, Denmark
trold@daimi.au.dk

ABSTRACT

We analyze a heads-up no-limit Texas Hold'em poker tournament with a fixed small blind of 300 chips, a fixed big blind of 600 chips and a total amount of 8000 chips on the table (until recently, these parameters defined the heads-up endgame of sit-n-go tournaments on the popular PartyPoker.com online poker site). Due to the size of this game, a computation of an optimal (i.e. minimax) strategy for the game is completely infeasible. However, combining an algorithm due to Koller, Megiddo and von Stengel with concepts of Everett and suggestions of Sklansky, we compute an optimal *jam/fold* strategy, i.e. a strategy that would be optimal if any bet made by the player playing by the strategy (but not bets of his opponent) had to be his entire stack. Our computations establish that the computed strategy is near-optimal for the unrestricted tournament (i.e., with post-flop play being allowed) in the rigorous sense that a player playing by the computed strategy will win the tournament with a probability within 1.4 percentage points of the probability that an optimal strategy (allowing post-flop play) would give.

Categories and Subject Descriptors

I.2.1 [**Artificial Intelligence**]: Applications and Expert Systems

General Terms

Algorithms, Theory

Keywords

Game playing, Game theory, Poker, Tournament

1. INTRODUCTION

In this paper, we describe the computation of a near-optimal (in a rigorous sense) strategy for a heads-up no-limit Texas Hold'em poker tournament with a fixed small blind of SB=300 chips, a fixed big blind of BB=600 chips and a total amount of 8000 chips on the table, and present and discuss the computed strategy. These exact parameters were chosen as they until recently[1] defined the heads-up endgame of the \$5 through \$30 buy-in sit-n-go tournaments on the popular PartyPoker.com online poker site.

We briefly review the rules of such a tournament (and of no-limit Texas Hold'em in general). The tournament is played between two players, Player 1 and Player 2. When the tournament starts, Player 1 receives s_1 chips and Player 2 receives s_2 chips where $s_1 + s_2 = 8000$. We want to keep s_1 and s_2 as parameters as the heads-up tournament we consider may be the endgame of a tournament with more people, so the two players should be able to enter the game with different stack sizes. The total number of chips on the table will remain 8000 for the duration of the tournament. The tournament consists of several *hands* of play. All chips bet by the players during a hand are put into the *pot*, to be won by whomever wins the hand. A hand is initiated by each player receiving two cards, hidden from the other player. A marker, known as the *button*, shows which of the players is to act first. After each hand, the button moves to the other player. Each hand starts with the players making two forced bets, known as the *small blind* and the *big blind* respectively, with the player with the button posting the small blind. Adopting popular terminology, we will sometimes refer to the *player* posting the small blind as "the small blind" and the player posting the big blind as "the big blind" when this can cause no confusion. Posting the blinds initiates the first of four *betting rounds.* In a betting round, the players must agree on how much to put into the pot to continue. The player to act has one of four options:

- *Fold* and forfeit hand. The other player wins the pot.
- *Call* and match the number of chips bet so far.
- *Raise* and bet additional chips beyond what is needed for a call.
- *Move all-in* and put all his chips (his stack) into the pot. The player has this option even if his stack is not big enough to make a call. In any case, the player remains in the hand without taking further actions. If he wins the hand, he can claim no more from the other player than what he put into the pot himself. Any unclaimed chips are returned to the other player after the hand.

[1]PartyPoker.com changed its blind structure on February 16, 2006 (unfortunately before we were able to make a profit from this research...)

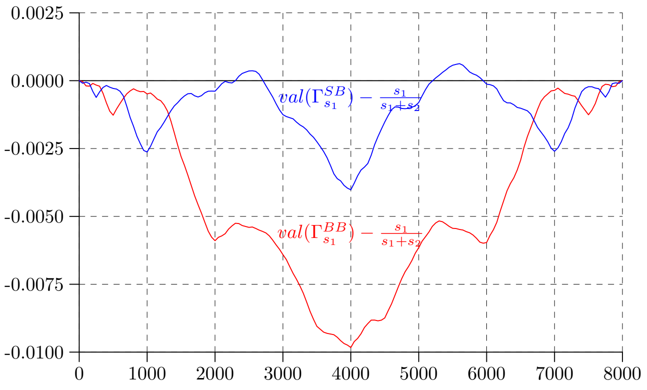

Figure 1: Difference between winning probability and relative stack size as function of the stack size.

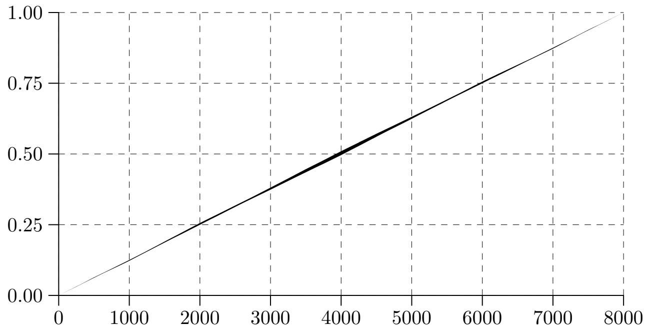

Figure 2: Area between upper and lower bound on the optimal winning probability of the player about to post the small blind as a function of his stack size.

The betting round continues until both players decline to make another raise. After this first betting round, three *community cards* are placed face-up, visible to both players. This is known as *the flop.* A second betting round then begins, started by the player *not* holding the button, but with no forced action. Then two more community cards are turned, with a third and fourth betting round after each. If both players remain after the fourth betting round, they each construct the best possible five card poker hand (according to standard poker rules) by selecting five cards among their two hidden cards and the five community cards. The highest hand wins the pot, or in case of a tie, the pot is split. If a player runs out of chips after play of a hand, he has lost the tournament and the other player has won.

In the rest of the paper, the tournament just described will be denoted by Γ. Before considering analyzing this game, one should note that there is no guarantee that the game will ever end; if both players keep folding any cards they are dealt, they will be in the exact same situation every other hand. Thus, the proper terminology and toolbox for analyzing Γ is Everett's notion of *recursive games* [8]. In the terminology of Everett, the game Γ should be viewed as decomposed into a set of *game elements*, each describing hands played with a particular distribution of chips and a particular position of the button. We use the standard convention of zero-sum games of letting a numeric payoff denote the gain of one distinguished player (Player 1) and hence the loss of the other player (Player 2), so the payoff models the losing player paying the winning a fixed amount. There are two categories of outcomes of playing a game element: First, the game element can terminate with a payoff of 1 being given to one of the players, a payoff of 0 to the other and no further action taken. This occurs when either player has run out of chips, and the tournament ends. The other possibility is for the game to continue at another game element in the set, in which case there is no payoff directly associated with the outcome. This happens when a player wins some, but not all of the chips of his opponent. Everett shows that the game elements in a recursive game can be assigned *critical values* that are analogous to the minimax values of games guaranteed to terminate (some subtle differences are described in Section 3). The critical value of a game element describes the expected payoff that Player 1 is guaranteed by playing optimally, so with the payoffs described above, the critical value of a game element is the probability of Player 1 winning the tournament, assuming optimal play. It is this vector of critical values (and hence, a vector of optimal winning probabilities) that we rigorously approximate in this paper by computing and formulate aconcrete strategy for Γ.

Our results on the approximation of the critical vector is summarized in Figure 1, showing the winning probability $val(\Gamma^{SB}_{s_1})$ guaranteed by the computed strategy for Player 1 when he is about to post the small blind as well as his guaranteed winning probability $val(\Gamma^{BB}_{s_1})$ when he is about to post the big blind, as a function of his stack size s_1. To improve readability, we have subtracted the relative stack size $(s_1/8000)$ from the computed winning probabilities.

In his seminal work on tournament poker, Sklansky [14, page 104-106] gives a heuristic argument that the optimal winning probabilities in a heads-up tournament are well-approximated by these values and the graph confirms this. That the estimate is not exact can be seen from the graphs. For instance, when the player about to post the small blind has a stack size of 2500, the computed strategy has a guaranteed winning probability strictly greater than 2500/8000, and of course, the *optimal* strategy for Γ would have an even bigger winning probability.

To estimate how much the strategy can be improved, we observe how well Player 1 could do if Player 2 adopts Player 1's strategy, thus giving an upper bound on the possible improvement. As shown by the graphs, $val(\Gamma^{SB}_{s_1})$ and $val(\Gamma^{BB}_{8000-s_1})$ sum up to more than 0.986 for any stack size s_1, leaving only room for a 0.014 improvement by any strategy. This is further illustrated in Figure 2, where both lower and upper bounds on the guaranteed winning probability of the player about to post the small blinds are plotted, as a function of his stack size. The upper bound is computed by subtracting from 1 the guaranteed winning probability of the computed strategy for the player about to post the big blind. The optimal strategy for Γ has a guaranteed winning probability somewhere in the narrow black area between the two curves. We define the *gap* of our approximation as the difference between the upper bound and the lower bound of the winning probabilities of the player about to post the small blind, i.e., the width of the black area in Figure 2. The size of this gap is depicted in Figure 3.

It is interesting to compare our analysis with some previous analyses of various versions of poker. Billings *et al* [2] focused on limit Texas Hold'em poker. The full game

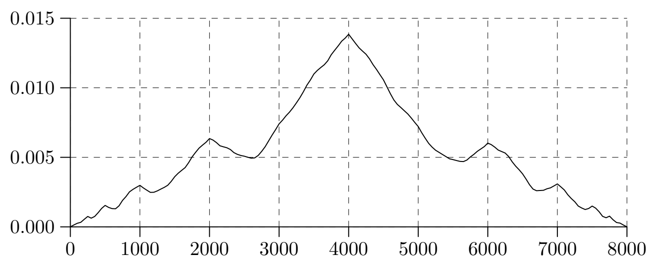

Figure 3: The gap of the computed strategy as a function of s_1.

tree for this game is far too large ($\approx 10^{18}$ nodes) to analyze completely. Instead, a hand-built *abstraction* of the game was analyzed. The abstraction was heuristically assumed to approximate the original game, but not in any rigorous sense. Gilpin and Sandholm [10] introduced a technique for automatically and exhaustively identifying rigorously sound equilibrium-preserving abstractions of the game tree for a class of well-structured games, including variants of poker. They successfully applied the technique to find optimal strategies [9] for *Rhode Island Hold'em* [13], which, however, is a significantly simplified version of Texas Hold'em. The automated abstraction technique also gave the option of allowing unsound abstractions, given that the loss incurred was below a given threshold. The latter technique was used to give a new heuristical abstraction [11] to the first three rounds of limit Texas Hold'em, as well as for the two last rounds separately. The threshold implied a rigorous guarantee on the size of the error, but unfortunately only to a three round pruning of the game, and therefore not for the full game. Andersson [1] computed an abstraction for no-limit Texas Hold'em, but again neither for the full game nor with any rigorous guarantees on the performance.

A common feature of these analyses is that they consider each hand as a *separate* game with the goal being to maximize the number of chips won in *each* hand, thereby aiming to be optimal in a *cash game*, rather than a tournament. In contrast to these works, we analyze a tournament. Compared to the works on Texas Hold'em, we obtain a strategy that *rigorously* approximates the optimal one (with a proven upper bound of 0.014 on the approximation error). In contrast to the work of Gilpin and Sandholm on Rhode Island Hold'em, we analyze a variant of poker that is being played in real life (and which is, in fact, the most popular variant of poker worldwide today). That it turns out to be possible to analyze the large game is primarily due to the fact that we consider *no-limit* rather than *limit* Texas Hold'em. Indeed, it does not seem computationally feasible with current technology to give a similarly proven near-optimal strategy to a *limit* Texas Hold'em tournament with the same stack sizes and blind level. One might naively expect no-limit Hold'em to be harder to analyze than limit Hold'em due to the fact that the game tree of a single hand is much larger: The fanout of the tree at a node where a player is to bet is roughly the number of chips at the disposal of that player. In contrast, in limit Hold'em, the player can choose only between the three options of calling, raising and folding. However, in the game Γ, the total number of chips is less than 14 times the big blind. This means that one of the players must have a *short stack*. Sklansky [14, page 119-121] notes that short-stacked no-limit poker is not as hard as it looks. In fact, even the completely trivial strategy of going all-in on every hand is a decent approximation of the optimum strategy. Indeed, the game where the strategy of Player 1 (the player aiming to maximize payoff) has been fixed to always going all-in has a critical vector which is a lower bound on the critical vector of the unrestricted game. Fixing the strategy for Player 2 (the player aiming to minimize payoff) in the analogous way gives a game whose critical vector is an *upper* bound on the critical vector of the unrestricted game. These lower and upper bounds are easily computed (their computation is a simple special case of the method described in Section 3), and in the case of Γ, such computations show the difference between the upper and lower bound to be no more than 5.3 percentage points in winning probability (Sklansky [14, page 120] reports a similar calculation for a tournament with different blind sizes).

We aim to achieve a strategy with worst-case behavior even closer to optimal. This can be achieved by sometimes folding a hand. Sklansky [14, page 122-127] designed a simple heuristic strategy known as "The System", which is easy to memorize even by people who have never played a hand of poker in their life. It simply specifies which hands one should go all-in with, and which ones one should fold, depending on whether anyone has raised so far in the hand. There are separate instructions for the big blind, but the strategy otherwise ignores position and stack size completely. Sklansky later devised a more sophisticated "revised system" [15], taking blind size and number of opponents into account. It is interesting to understand how far such "systems" can be pushed. Sklansky writes [14, page 126]: *I am extremely curious as to how strong it* [The System] *really might be. And if it is strong, how much stronger yet a more complex, move all-in, system would be?* This paper studies Sklansky's question for the game Γ. Adopting terminology that has become increasingly popular, we define a *jam/fold* strategy to be a strategy for which any bet is the entire stack. Thus, "The System" and its revision are jam/fold strategies. The strategy we compute is the *optimum* (in the minimax sense) jam/fold strategy for Γ and thus answers Sklansky's question for this particular game and assuming a worst case opponent (the game-theoretic setup). Our graphs above show it to be quite strong, losing less than 1.4 percentage points (and for many stack sizes significantly less) winning probability compared to the optimal unrestricted strategy, i.e., an upper bound significantly smaller that the corresponding figure of 5.3 percentage points for the "always-jam" strategy (it is interesting to note that for the parameters of Γ, the revised "system" is in fact equivalent to "always-jam").

As for the case of "always-jam", the reason computing optimal jam/fold strategies turns out to be feasible is that they make analysis of the post-flop game irrelevant. If either player follows a jam/fold strategy, the post-flop game will only be reached if both players are all-in, in which case there are no further actions by the players in the hand. As a consequence, the entire post-flop game consist of random choices by nature, and can therefore be collapsed into an expected payoff which can be straightforwardly computed given the hidden pocket cards of the two players.

In the remainder of the paper, we describe the computation and the computed strategy in more detail.

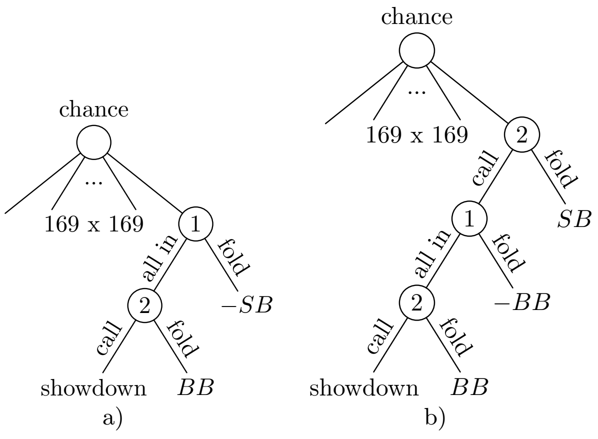

Figure 4: Game trees for a single hand if Player 1 is restricted to jam/fold. To the left is the case where Player 1 posts small blind, and to the right is the case where Player 1 posts big blind.

2. ANALYZING HANDS

We first consider the analysis of the play of *single hands* and then in the next section show how to generalize the considerations to the tournament setting. We will restrict the strategy space of Player 1 to jam/fold strategies when he posts the small blind as well as when he posts the big blind.

To find the best jam/fold strategy when Player 1 posts the small blind, we simply restrict his actions to these two possibilities. His opponent will now only have to consider when to call an all-in bet and when to fold. Since each player can see only two cards, only the values of these cards and whether they are of the same suit or not are relevant parameters for the decision. This reduction leads to 169 different combinations of pocket cards for each player, and thereby to the game tree sketched in Fig.4. Note that information sets are not shown, but that each player has 169 information sets; one for each possible pocket hand. Since the game is defined by restricting the action space of Player 1, its value is a lower bound on the value of a hand for Player 1 when no such restrictions are made. Koller, Megiddo and von Stengel [12] showed how to compute optimal strategies for a game with hidden information by solving a linear program of size proportional to the game tree. The size of the tree of Fig.4 is sufficiently small for this algorithm to be successfully applied. Previous to our work, Robert Anderson (personal communication) made such a computation for various settings of the parameters and discussed his findings in the forums on the website `www.twoplustwo.com`. This part of our computation is essentially a reproduction of his computations (and the results have been verified as consistent). Using the linear programming solver PCx [6], we computed values and minimax strategies for all hands with parameters $SB = 300$, $BB = 600$ and combined stack size 8000, for all individual stack sizes being multiples of 50. To get rigorous bounds, we wanted exact rational solutions to the linear programs. PCx, on the other hand, provided floating point solutions. To get exact solutions, we used a technique similar to that of Dhiflaoui *et al* [7]: We interpreted the floating point solution as an exact description of an approximately optimal solution and used a short sequence of pivotings using exact rational arithmetic to obtain an optimal and exact rational tableau for the linear program. We verified optimality using linear programming duality, which, for the linear programs at hand, amounts to computing an optimal counter-strategy against the strategy computed for each of the two players and checking that the payoffs obtained by the counter-strategies match. The strategies computed were pure[2] on most hands, but not completely pure. A list of hands mixed by the small blind (i.e., folded with non-zero probability and jammed with non-zero probability) for different stack sizes is given as the table of Figure 5. The computed strategy for the case of both stacks being of size 4000 can be seen as Figure 6a. All the computed strategies (also those mentioned later in the paper) can be found at `www.daimi.au.dk/~bromille/pokerdata/`.

Stack	Mix	Stack	Mix	Stack	Mix
1800	T2	3550	63s	4600	65
2100	95	3600	86	4650	63s
2200	92s	3650	94s	4700	T6
2250	92s	3700	63s	4750	J4
2550	T5	3750	86	4900	96
2600	43s	3800	94s	4950	96
2700	93s	3850	T2s	5000	96
2800	93s	3950	86	5100	43s
2850	93s	4000	86	5150	93s
2900	43s	4050	86	5200	93s
3000	96	4150	T2s	5300	93s
3050	96	4200	94s	5400	43s
3100	96	4250	86	5450	T5
3250	J4	4300	63s	5750	92s
3300	T6	4350	94s	5800	92s
3350	63s	4400	86	5900	95
3400	65	4450	63s	6200	T2
3500	94s	4500	94s		

Figure 5: Mixed hands for the player posting small blind as a function of his stack size.

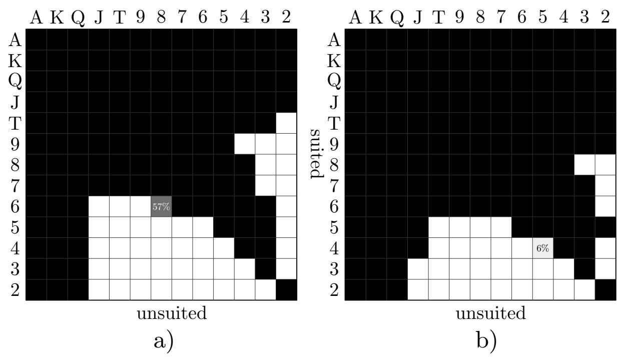

Figure 6: Computed strategy for the single-hand play when $s_1 = s_2 = 4000$. On the left is the strategy for the player posting small blind, on the right is the reply of the player posting big blind. White means fold, black mean jam.

Streib [16] reports a computation somewhat similar to ours and those of Robert Anderson but with some important differences and it is thus interesting to compare his results with ours. First, Streib considered a game where the action space of Player 1 is restricted to jam/fold as in our game of Figure 4, but where the action spaces of the players are fur-

[2]Prescribed a deterministic action.

stacksize	43s	J2
1800	fold	jam
3600	jam	fold

Figure 7: Unique minimax jam/fold actions for the small blind.

ther restricted to calling with some percentage of their *top hands* according to certain *hand rankings*. It is not clear to us that analyzing such a game gives a rigorous lower bound on the actual value of the hand in the same way that our analysis does. Interestingly, Streib found *pure* equilibria for his game for all parameters considered (using an iterative procedure rather than linear programming), while our computations establish that for the game of Fig. 4 and many values of the parameters, there *are* no pure equilibria, even though we found no case where more than one hand needs to be mixed. In particular, using the results of our computations we can show the following (somewhat perplexing) theorem.

THEOREM 1. Any *optimal (in minimax sense) jam/fold strategy for the small blind in a single hand of parameters SB = 300, BB = 600 and stack sizes of both players being 4000* must *mix play of 86 offsuit (i.e., must fold this hand with non-zero probability and jam it with non-zero probability). On the other hand, a minimax jam/fold strategy for the small blind mixing* only *the play of 86 offsuit exists.*

PROOF. We actually give a "meta-proof" showing how a proof can be given based on our computations.

One could, in theory, try all 2^{169} pure strategies, and verify that none of them obtains the value of the game against an optimal counter-strategy, which can easily be computed. Trying all of them is of course infeasible, but we can use the computed strategy σ for Player 2 to rule out most of them. If a pure strategy is a minimax strategy, it must be a pure best reply to σ. This means that it must choose the action with the highest expected value against σ for all possible hands, thus fixing the required action for most hands. In fact, there is only one hand, 86 offsuit, where the expected payoff against σ is the same for both actions. Thus we only have to check two pure strategies, and it turns out that none of them obtains the value of the game against an optimal counter-strategy. This calculation was done using exact rational arithmetic to make the present proof rigorous. □

The procedure outlined in the proof above can be easily generalized to check for pure minimax strategies in arbitrary two-player zero-sum extensive-form games with perfect recall. The general version will use time exponential in the number of non-pure moves in the computed (mixed) minimax strategy (found using the Koller-Megiddo-von Stengel algorithm). One cannot hope for a polynomial-time algorithm, as the problem of deciding whether a given two-player zero-sum extensive-form game with perfect recall has a pure equilibrium is **NP**-hard [3].

Another interesting result of our computation is given in the table of Figure 7. The table shows the action of the small blind on hands 43 suited and J2 unsuited with a stack size of 1800 and 3600. The computed strategy folds 43s and jams J2 with the smaller stack and vice versa with the bigger stack. Furthermore, using the technique also used to prove Theorem 1, we have established that this pattern occurs in *all* minimax jam/fold strategies for the small blind with the stated stack sizes, not just in the computed strategy. An intuitive explanation for the pattern is the following: With the small stack size, the big blind will call an all-in bet with the vast majority of his hands, including several "trash" hands containing two medium sized cards. 43s fares badly against these hands while J2 does okay, due to the "highish" jack. On the other hand, with the bigger stack, the average quality of a calling hand is higher. In particular, it is more likely to include a jack and 43s is thus less likely to be dominated by such a hand than J2 is. *The pattern shows that it is impossible to devise a fixed hand ranking valid for all stack sizes so that low-ranked hands are folded and high-ranked hands are jammed in the optimal jam/fold strategies.*

We have described how to analyze jam/fold strategies for Player 1 when he posts the small blind. The situation is a little more complicated when we instead want to analyze jam/fold strategies for Player 1 in a hand G where he posts the big blind. It might be tempting to simply use the strategy for the big blind computed for the game of Figure 4, but doing so would be a fallacy: We, as Player 1, want to assume that we are restricted to jam or fold but we should make no such assumptions about our opponent. In particular, we *cannot* assume that our opponent, when he moves all-in, had no other options besides folding. To analyze a hand where Player 1 is posting the big blind, the game of interest is a game G' where Player 1 is restricted to jam/fold, but Player 2 (posting the small blind and acting first) is still allowed the possibility to bet any amount he wishes. This makes the value of G' a lower bound for the value of G. However, the many possible actions of Player 2 makes G' infeasible to analyze directly. To solve this problem, it turns out to be useful to analyze a game G'' where the actions of Player 2 as well as Player 1 have been restricted. To define G'', we restrict the actions of Player 2 (posting the small blind and acting first) to {call, fold}, and the actions of Player 1 (posting the big blind and acting second) to {all-in, fold} obtaining the game tree sketched in Figure 4. The following lemma establishes that we can analyze G'' instead of G'.

LEMMA 2. *The value of G' is equal to the value of G''. Furthermore, an optimal (minimax) strategy for Player 1 in G'' can be converted to an optimal (minimax) strategy for Player 1 in G' in the following way: No matter what Player 2 does (except fold), Player 1 responds as he would against a call by Player 2 in G''.*

PROOF. For the zero-sum game G', pairs of minimax and maximin strategies correspond exactly to Nash equilibria of G'. Also, note that the optimal strategy for Player 2 from G'' can be immediately interpreted as a strategy in G'. Hence, we just have to show that the strategy stated for Player 1 is in equilibrium with the optimal strategy from G'' for Player 2, even if played in G'. So consider these strategies played against each other. Player 1's strategy is clearly a best response to Player 2's, since Player 2 only uses the actions also available to him in G'' and the actions available to Player 1 are the same in the two games. Thus, we just have to show that Player 2's strategy is a best response to Player 1's. But Player 1 responds in the same way to any non-folding action made by Player 2. The size of the bet made by Player 2 only influences the final payoff if Player 1 goes all-in and Player 2 subsequently folds. Hence, Player

2's best response is to make such a bet as small as possible, and should in fact just call. Thus, his optimal strategy from G'' is still a best response to Player 1's strategy, also in G', as was to be proved. $\square$

Note that Lemma 2 tells us that the big blind, playing a minimax jam/fold strategy can completely *ignore* the action of the small blind. The size of G'' is sufficiently small to be analyzed using the Koller-Megiddo-von Stengel algorithm. We did this for the same range of parameters as for the case of the small blind, again using the PCx linear programming solver. The computed strategy for the case of both stacks being of size 4000 can be seen as Figure 6b. Note that the figure prescribes *looser* play than Figure 6a. This is quite natural since Player 1 ignores the action of Player 2 (posting the small blind) and simply assumes that Player 2 called. Under that assumption, the pot is now larger than in the corresponding situation for the small blind in the game of Figure 4, resulting in looser play. In contrast, the optimal strategy for Player 2 (posting the big blind) in the game of Figure 4 is *tighter* than the strategy of Figure 6a. But as already noted, the optimal strategy for Player 2 in the game of Figure 4 is irrelevant for us.

Note that the computed strategy may fold after a call of the small blind, even though the flop could be seen for free. This is a necessary consequence of the restriction to jam/fold strategies (*computationally*, we cannot afford to see the flop!). It is possible to get a computationally feasible strategy which does marginally better than the computed one and avoids folding after a call of the other player as follows: Instead of folding after a call, we check. At any point in time in the rest of the hand (i.e., postflop) when we are supposed to act, we compute whether we have a hand which against *any* possible hand of the opponent has a better expected payoff from a jam (and a call from the opponent and thus a showdown) than from a fold. If we do, we go all-in. If we don't, we fold, unless we can check. The required computation could be done on-the-fly very efficiently, as the flop is fixed.

3. ANALYZING THE TOURNAMENT

In the previous section, we described the analysis of *single hands*. However, as pointed out by Sklansky [14, page 19-22], correct play in a tournament situation may differ significantly from correct play in a cash game situation. Sklansky's arguments apply even for heads-up tournaments and assuming optimal players, i.e., for the game-theoretic setup considered in this paper. In this setup, we may rephrase Sklansky's argument as follows: In a cash game, a player may choose to leave the table. In a tournament situation, if the tournament doesn't end with the current hand, he is forced to play another hand. This other hand may have a negative expected payoff for him (for instance, he may have to post the big blind while the small blind has the advantage with the given parameters). If it does have a negative expected payoff, he is better off trying to avoid playing it and this will influence his optimal strategy in the current hand towards trying to *end* the tournament, typically by playing somewhat looser. Conversely, if the next hand has positive expected payoff for him, he'll play the current hand tighter.

In this section, we consider how to modify the analysis of single hands to an analysis of the entire tournament. This also enables us to give a precise quantitative insight in the difference between optimal play in the tournament Γ and the corresponding single hands. We intend to compute a strategy for Player 1 who posts the small blind every other hand and the big blind every other hand by restricting him to jam/fold-strategies in both situations, using Everett's notion of recursive games explained in the introduction. To interpret the tournament as a recursive game, we define a game element for each possible configuration that can occur before the cards are dealt in a hand. Each game element has the form of one of the games from Fig.4, but instead of each payoff, there is a reference to another game element. More formally, let $C = 8000$ be the total number of chips in play. We define Γ^i_k as the game element where Player 1 has k chips and Player 2 has the remaining $C - k$ chips, and Player i has the dealer button (Γ^1_k was denoted Γ^{SB}_k in the introduction, while Γ^2_k was denoted Γ^{BB}_k). In the following, index j is used to indicate the opponent of Player i, i.e., $j = 3-i$. If a hand is played in game element Γ^i_k, and Player 1 wins c chips, the game continues at game element Γ^j_{k+c}. If game element Γ^i_C or Γ^i_0 is encountered, the game ends with Player 1 or 2 as the respective winners. We let payoffs in these sinks be 1 if Player 1 wins, and 0 if he loses. For practical reasons, we don't want to consider game elements Γ^i_k for *every* $k \in \{0, 1, 2, \ldots, 8000\}$. We observe that if k is a multiple of 50 it stays that way, so we get a subgame by considering only $k \in I = \{0, 50, 100, \ldots, 8000\}$. A strategy for this subgame can be extended to arbitrary stack sizes by simply ignoring at most 49 chips. Clearly, any computed lower bound on the guaranteed winning probability will still be valid. Thus, the number of non-trivial game elements we actually consider is only $2 \cdot (\#I - 2) = 318$.

To analyze the recursive game just defined, we need to explain in more detail notions from Everett's theory of recursive games [8]. A *stationary strategy* of a recursive game is a collection of strategies, one for each game element, where the strategy associated with a given game element is to be used whenever that game element is encountered during play. For instance, for the game Γ, a stationary strategy is one that plays a hand with a given stack distribution and button position in the same (possibly mixed) way every time such a hand is encountered. For general zero-sum recursive games, Everett shows the existence of a real-valued *critical vector* (v_g) with one entry for each game element g, so that

- For any $\epsilon > 0$, there is a stationary strategy for Player 1 so that for all game elements g, if play starts in g and if Player 1 plays by the strategy, he is guaranteed an expected payoff of at least $v_g - \epsilon$,
- For any $\epsilon > 0$, there is a stationary strategy for Player 2 so that for all game elements g, if play starts in g and if Player 2 plays by the strategy, Player 2 is guaranteed an expected loss of no more than $v_g + \epsilon$.

In both cases, we refer to the strategies as *ϵ-optimal.* Since $\epsilon > 0$ can be made arbitrarily small in the above statements, they allow us to think of an entry of v_g as analogous to the usual value of a zero-sum game. However, it is worth pointing out that strictly speaking, recursive games may not have minimax (optimal) strategies, i.e, even though strategies achieving expected payoff *arbitrarily close* to v_g exists, expected payoff *exactly* v_g may not be attained by any stationary strategy. Indeed, Everett presents simple games with this property and we do not have any proof that the game Γ considered here is any different. Thus, our claim above of having computed "the optimal jam/fold strategy" is actually slightly misleading; we really only aim to compute a stationary strategy achieving the critical values for the jam/fold game elements *minus* a very small ϵ (corresponding in our concrete calculations to a probability mass of roughly 10^{-8}). But since we aim only to use these strategies to approximate the critical vector for the unrestricted game anyway, this point is largely irrelevant.

Recursive games generalize *simple stochastic games* and no efficient algorithm is known for analyzing that class of games [4, 5]. However, Everett's existence proof of the critical vector is accompanied by the following *non-deterministic* algorithm for computing a stationary strategy for Player 1 with a particular performance guarantee (and a *lower* bound for the critical vector) and a stationary strategy for Player 2 (and an *upper* bound for the critical vector). Everett's non-deterministic algorithm is based on the following notions. Given a vector of values with one entry for each game element, one can reduce a game element to an ordinary (non-recursive) game by replacing the outcomes leading to other game elements with their respective entry in the given value vector. If all game elements are evaluated with respect to the given value vector, a new value vector is obtained. This evaluation defines a map $\mathbb{M}$, called the *value map.* The critical vector of the game is a fixed point of the value map, but it may not be the only one. Let $\preceq$ and $\succeq$ be partial orders on vectors of real numbers (one entry for each game element), defined by:

$$\vec{U} \succeq \vec{V} \Leftrightarrow \left\{ \begin{array}{ll} U^i > V^i & \text{if } V^i > 0 \\ U^i \geq V^i & \text{if } V^i \leq 0 \end{array} \right\} \forall i$$

$$\vec{U} \preceq \vec{V} \Leftrightarrow \left\{ \begin{array}{ll} U^i < V^i & \text{if } V^i < 0 \\ U^i \leq V^i & \text{if } V^i \geq 0 \end{array} \right\} \forall i$$

Intuitively, 0 is a special case in the above definition as it is the payoff of infinite play.

THEOREM 3 (EVERETT, 1957).

- *If $\mathbb{M}(v) \succeq v$, then v is a lower bound on the critical vector. Furthermore, the stationary strategy for Player 1 obtained by finding the optimal strategy in each game element, with arcs to other game elements replaced by the corresponding values in v, has guaranteed expected payoff* at least *v_g for play starting in g.*
- *If $\mathbb{M}(v) \preceq v$, then v is an upper bound on the critical vector.*

Thus, if we manage to guess $v_1 < v_2$ so that $\mathbb{M}(v_1) \succeq v_1$ and $\mathbb{M}(v_2) \preceq v_2$ and so that the entry-wise difference between v_2 and v_1 is at most ϵ, we have approximated the critical vector with additive error at most ϵ. In our case, the following simple iterative procedure provided such a good guess for the critical vector, with additive error at most around 10^{-8}. We initially defined all entries of v_1 to be 0 and all entries of v_2 to be 1 and then iterated the assignments $v_1 := \mathbb{M}(v_1)$ and $v_2 := \mathbb{M}(v_2)$ until v_1 and v_2 were sufficiently close or until the conditions $\mathbb{M}(v_1) \succeq v_1$ and $\mathbb{M}(v_2) \preceq v_2$ stopped being valid. For many games, this iteration would *not* converge to the critical vector, but in our case, it did. A single iteration of $v_i := \mathbb{M}(v_i)$ meant solving 318 linear programs, each containing between 500 and 1000 constraints and variables. For this computation we used a

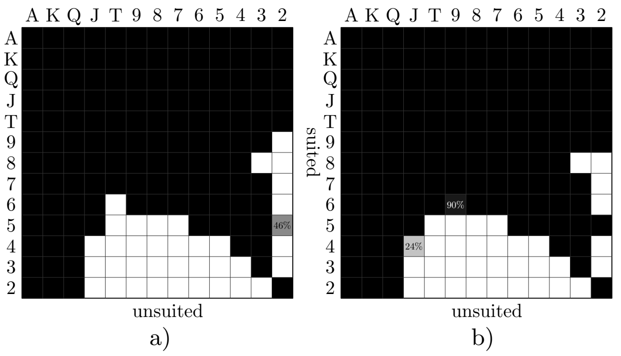

Figure 8: Computed strategy in the tournament when $s_1 = s_2 = 4000$. On the left is the strategy for the player posting small blind, on the right is the reply of the player posting big blind.

cluster of twelve Linux machines running the PCx linear programming code to solve around 20,000 linear programs, each requiring around a minute for computation and verification of equilibrium, totaling roughly two weeks of CPU-time.

The computed strategies were purified in the same way as the single hand strategies of the last section. Some quantitative properties of the computed strategy were already discussed in the introduction.

As stated earlier in the paper, Sklansky gave informal arguments for the following two statements about tournament play:

1. Correct play in a tournament situation may differ significantly from correct play in a cash game situation, even heads-up, assuming optimal players, with blinds and stack sizes in the tournament situation matching those in the cash game situation.
2. When Player 1 has a stack of size s_1 chips and Player 2 has a stack of size s_2 chips, then, assuming optimal play, Player 1 wins with probability *approximately* $s_1/(s_1 + s_2)$.

We already rephrased Sklansky's argument for Statement 1. We may rephrase his argument for Statement 2 as follows: If we interpret the initial stack sizes stated as the buy-ins of the players, then, since Player 1 goes to the table with s_1 chips and Player 2 goes to the table with s_2 chips and both play optimally, the expected payoff from the tournament for Player 1 is s_1 chips, as he can expect to take the same amount out of the game as he brought in. Hence, he must win the prize (i.e., s_1+s_2 chips) with probability $s_1/(s_1+s_2)$.

Note that this informal argument for Statement 2 ignores the *asymmetry* of the situation: That one of the players will post the first big blind and the other the first small blind and that this may give one of the players (we may not know which *a priori*) an advantage not considered in the argument. Hence, the stated probability is only approximately right. The situation is further complicated by the fact that it might not be the same blind that has the advantage for different stack sizes.

Note that it is precisely the asymmetry of the game that make Statement 1 true and Statement 2 only approximately true. Thus, intuitively, the more significant Statement 1 actually is, the less precise Statement 2 is. Our computations give us some precise quantitative insight about the significance of the first statement and the precision of the last for the concrete case of Γ. Figure 2 shows that Statement 2 is in fact a fairly accurate statement. To gauge the significance of Statement 1, we fixed the strategies of Player 1 in the game elements to the single-hand jam/fold strategies computed in Section 2 and computed the optimal counter-strategy of Player 2 in the recursive game. This made the probabilities of winning for Player 1 drop by less than 0.1 percentage points compared to the winning probabilities of his optimal jam/fold strategies in the game elements in the recursive game. This insignificant drop indicates that for Γ (or more precisely, for the jam/fold approximation to this game), the effect of Statement 1 is almost completely insignificant and one gets an extremely good approximation of correct play by playing the tournament hand-for-hand. However, it does not seem clear to us how to establish this *a priori* without actually analyzing the recursive game as we did.

An interesting observation is that the critical vector can be viewed as a utility function applied to the hand-for-hand game, where the utility of a given stacksize is the corresponding probability of winning. If we reexamine the red graph of Figure 1, being the small blind's utility function, we can see that this utility function implies a slight risk tolerance for many stacksizes, e.g. $s_1 = s_2 = 4000$. This gives an intuitive explanation for why the strategy in Figure 8a prescribes going all-in slightly more often than the corresponding strategy in Figure 6a. This effect is not noticeable for the case of the big blind, since the utility function is much closer to being linear.

4. CONCLUSIONS AND OPEN PROBLEMS

We have computed a near-optimal strategy for a particular short-stacked no-limit Hold'em tournament. While short-stacked no-limit Hold'em is arguably an easier game than limit Hold'em, this is not the case for *deep-stacked* no-limit Hold'em as pointed out by several experts. Indeed, unlike (say) chess, no-limit Hold'em is not a finitary game as we can consider the game as we let the stack size approach infinity. When we do, minimax (or just approximate minimax) play becomes a complete mystery to us. For instance, consider an unrestricted (i.e., not a jam/fold) single hand heads-up no-limit Hold'em hand with blinds of one and two and a stack size of, say $s = 10^{100}$ for both players. Note that the value of the game is between -1 and 2 for the small blind. We can do a little better by considering the optimal jam/fold strategy for the small blind which for this stack size consists of folding everything except aces and going all-in with aces. This strategy has a guaranteed expected payoff of $-1 + 3673/270725$, thus yielding a slightly better lower bound for the value of the game than -1. Similarly, the strategy for the big blind of folding everything except aces and going all-in with those in response to any bet by the small blind has a guaranteed expected payoff for the big blind of $-2 + 3674/270725$, thus yielding an upper bound of $2 - 3674/270725$ for the value of the game for the small blind. We know no better bounds and thus ask whether one can prove that it is *not* a minimax strategy (in the unrestricted game) for the small blind to fold everything except aces and go all-in with aces when the stack size s becomes sufficiently large? We certainly conjecture that the stated strategy is *not* minimax. But it seems that any proof would involve devising another strategy with a better guarantee than $-1 + 3673/270725$. Such a strategy would presumably have to play other hands than aces and would have to see a flop in some situations (as the small blind clearly cannot go all-in with any hand except the nuts with the stated stack size, and the big blind may call whichever non-all-in bet the small blind makes in any such situation). Giving non-trivial bounds for any such strategy seems utterly hopeless to us.

4.1 Acknowledgments

We are indebted to Robert Anderson who previously computed minimax jam/fold-strategies for single-hand play with Player 1 being the small blind and who pointed out an error in our computation. Also, we thank posters on the "1-table Tournaments" forum on www.twoplustwo.com who provided useful feedback to a previous version of this paper, in particular posters *the shadow* and *marv*.

5. REFERENCES

[1] R. Andersson. Pseudo-optimal Strategies in No-Limit Poker. Master's thesis, Umeå University, May 2006.

[2] D. Billings, N. Burch, A. Davidson, R. Holte, J. Schaeffer, T. Schauenberg, and D. Szafron. Approximating Game-Theoretic Optimal Strategies for Full-Scale Poker. In *IJCAI*, 2003.

[3] J. R. S. Blair, D. Mutchler, M. van Lent. Perfect Recall and Pruning in Games with Imperfect Information. *Computational Intelligence*, 12:131–154, 1996.

[4] A. Condon. The Complexity of Stochastic Games. *Information and Computation*, pages 203–224, 1992.

[5] A. Condon. On Algorithms for Simple Stochastic Games. *Advances in Computational Complexity Theory, DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, 13:51–73, 1993.

[6] J. Czyzyk, S. Mehrotra, M. Wagner, S. J. Wright. PCx: An Interior-Point Code for Linear Programming. *Optimization Methods and Software*, 12:397–430, 1999.

[7] M. Dhiflaoui, S. Funke, C. Kwappik, K. Mehlhorn, M. Seel, E. Schmer, R. Schulte, and D. Weber. Certifying and Repairing Solutions to Large LPs, How Good are LP-Solvers? In *SODA*, pages 255–256, 2003.

[8] H. Everett. Recursive games. In *Contributions to the Theory of Games Vol. III*, Vol 39 of *Annals of Mathematical Studies*, 1957.

[9] A. Gilpin and T. Sandholm. Optimal Rhode Island Hold'em Poker. In *AAAI*, 2005.

[10] A. Gilpin and T. Sandholm. Finding Equilibria in Large Sequential Games of Incomplete Information. In *Electronic Commerce*, pages 160–169, 2006.

[11] A. Gilpin and T. Sandholm. A Texas Hold'em poker player based on automated abstraction and real-time equilibrium computation. In *AAMAS*, 2006.

[12] D. Koller, N. Megiddo, and B. von Stengel. Fast Algorithms for Finding Randomized Strategies in Game Trees. In *STOC*, pages 750–759, 1994.

[13] J. Shi and M. L. Littman. Abstraction Methods for Game Theoretic Poker. In *Computers and Games*, pages 333–345, 2001.

[14] D. Sklansky. *Tournament Poker for Advanced Players.* Two Plus Two Publishing, 2002.

[15] D. Sklansky. *Card Player Magazine*, 2003.

[16] T. Streib. Blind versus Blind Play in Tournament Situations, Nov. 2005. Two Plus Two Internet Magazine (twoplustwo.com).