Computing an Approximate Jam/Fold Equilibrium for 3-player No-Limit Texas Hold'em Tournaments *

Sam Ganzfried
Department of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213
sganzfri@cs.cmu.edu

Tuomas Sandholm
Department of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213
sandholm@cs.cmu.edu

ABSTRACT

A recent paper computes near-optimal strategies for two-player no-limit Texas hold'em tournaments; however, the techniques used are unable to compute equilibrium strategies for tournaments with more than two players. Motivated by the widespread popularity of multiplayer tournaments and the observation that jam/fold strategies are near-optimal in the two player case, we develop an algorithm that computes approximate jam/fold equilibrium strategies in tournaments with three — and potentially even more — players. Our algorithm combines an extension of fictitious play to imperfect information games, an algorithm similar to value iteration for solving stochastic games, and a heuristic from the poker community known as the Independent Chip Model which we use as an initialization. Several ways of exploiting suit symmetries and the use of custom indexing schemes made the approach computationally feasible. Aside from the initialization and the restriction to jam/fold strategies, our high level algorithm makes no poker-specific assumptions and thus also applies to other multiplayer stochastic games of imperfect information.

Categories and Subject Descriptors

I.2 [**Computing Methodologies**]: Artificial Intelligence

General Terms

Algorithms, Economics

Keywords

Game theory, equilibrium finding, stochastic games, imperfect information, poker, Texas hold'em

1. INTRODUCTION

Poker exemplifies many challenging computational problems in multiagent systems: it is a game of imperfect information, its strategy spaces are extremely large, and there can be many players. There has been significant interest and progress recently in developing game theory based strategies for poker in AI. For example, the game of heads-up (i.e., two-player) Rhode Island hold'em has been solved [9]. Regarding Texas hold'em, there has been great progress in finding good game theory based solutions for playing a hand of heads-up limit Texas hold'em [2, 7, 8, 10], but relatively little work has been done on the no-limit variant[1] and tournament poker despite the fact that these are perhaps the most popular variants among humans. (In limit poker bets must be of a fixed size, while in no-limit poker players can bet any amount up to the amount of chips they have left.)

One significant exception is a recent paper that studies no-limit tournaments [13] and presents very interesting results using a restricted class of strategies called jam/fold strategies. However, the results and the algorithms of that paper do not apply to more than two players. This is not too surprising as there often exists a large complexity gap between the difficulty of solving two and three-player zero-sum games. For example, two-player zero-sum matrix games can be solved in polynomial time by linear programming while solving three-player zero-sum matrix games is PPAD-complete [4]. Furthermore, poker tournaments are stochastic games, and while it has been known for decades that an equilibrium exists in two-player zero-sum undiscounted stochastic games, it is still unknown whether one exists in such games with more than two players.

2. NO-LIMIT TEXAS HOLD'EM POKER

No-limit Texas hold'em is by far the most popular form of poker currently played, and we will briefly review the rules here. All players at the table are dealt two private *hole cards*, and one player is selected to be the *button* — a designation which shifts one position clockwise each hand. The player to the left of the button is called the small blind (SB), and the player to his left is the big blind (BB). Both the SB and BB are forced to put some number of chips into the pot before the hand (normally BB = 2 × SB); these investments are called *blinds*. Then there is a round of betting starting with the player to the BB's left. After that, three cards (called the *flop*) are dealt face up in the middle of the table. Then there is another round of betting starting with the player directly left of the button (this player begins all future betting rounds as well), followed by another card dealt face up (the *turn*). Then another round of betting, followed by a fifth card face up (the *river*), followed by a final round of betting. If two

*This material is based upon work supported by the National Science Foundation under ITR grant IIS-0427858. We also acknowledge Intel Corporation and IBM for their gifts.

Cite as: Computing an Approximate Jam/Fold Equilibrium for 3-player No-Limit Texas Hold'em Tournaments, Sam Ganzfried and Tuomas Sandholm, *Proc. of 7th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2008)*, Padgham, Parkes, Müller and Parsons (eds.), May, 12-16., 2008, Estoril, Portugal, pp. XXX-XXX.

[1] A notable exception is a contemporary paper which studies a single hand of heads-up no-limit Texas hold'em [11].

or more players remain in the hand after the final betting round, the player with the best five-card hand (constructed from his two hole cards and the five community cards) wins the pot. In case of a tie, the pot is split evenly among those players.

During each round of betting, each player has four possible options. (1) *fold*: pass and forfeit his chance of winning the pot. (2) *call*: put a number of chips equal to the size of the current bet into the pot. (3) *raise*: put additional chips in the pot beyond the amount needed to call. (4) *jam* (also referred to as moving *all-in* or *pushing*): put all of one's remaining chips into the pot (even if that amount is smaller than the amount needed to call; in this case a *side pot* is created for the other players so that everyone has invested equally into the main pot).

Sit-and-go tournaments are an extremely popular form of poker, especially on online poker sites. They work as follows. Some number of players (usually 9) all pay an entry fee (say \$15) and are given a number of chips (say 1500), which have no monetary value per se except that a player is eliminated from the tournament when he runs out of chips. The first six players eliminated from the tournament lose their entry fee, while the three top finishers receive 50%, 30%, and 20% of the sum of the entry fees (9×\$15) respectively. Usually the blinds increase every five or ten minutes, starting at SB = 10, BB = 20 and approximately doubling at every level. Since chips have no explicit monetary value, tournaments are actually stochastic games, where each *state* corresponds to a vector of stack sizes.

3. JAM/FOLD STRATEGIES AND HEADS-UP RESULTS

When blinds become sufficiently large relative to stack sizes, rationality dictates more aggressive play. For example, suppose a player has 1000 chips at the 100/200 level (SB = 100, BB = 200) and is considering making a preflop raise smaller than going all-in. If he makes a minimum raise to 400 and someone re-raises for all his chips, then he will only need to put in 600 more chips into a pot of 1700. If all chips had the same value, then he would only need to expect to win with probability $\frac{600}{1700+600} = 0.261$ to make calling correct. Since even the worst starting hand has close to this probability of beating the best starting hand, it will almost always be correct to call. If he had gone all-in preflop instead of raising to 400, the result would probably have been the same in this case; furthermore, he would have been more likely to win the blinds for free, since opponents are more likely to call a raise of 200 than a raise of 800.

In line with this example, common strategy when blinds become sufficiently high relative to stack sizes is to either go all-in or fold preflop (this is known as a *jam/fold strategy*). Following this reasoning, a recent paper [13] computes near-optimal jam/fold strategies for tournaments with two players and the fixed parameters SB = 300, BB = 600, and 8000 total chips (at the time, these defined the normal parameters for the heads-up endgame in tournaments on PartyPoker.com). In fact, if we let G_{i,s_i} denote the restriction of the original tournament in which player i starts with s_i chips and is limited to playing a jam/fold strategy (while the other player is not), they show that for any value of s_1, $V_1(G_{1,s_1}) + V_2(G_{2,8000-s_1}) \geq 0.986$ (where the winner gets payoff 1, the loser gets payoff 0, and V_i denotes the value of the game to player i). Thus, neither player can guarantee more than 0.014 by deviating to a non-jam/fold strategy; this provides a justification for restricting attention to jam/fold strategies. They compute the optimal jam/fold strategies for the full stochastic games G_{1,s_1} and G_{2,s_2}, where they consider all states in which stack sizes are a multiple of 50. One important conclusion they draw is that for any starting stack sizes s_1, s_2, the probability that player i will win is very close to

$$\frac{s_i}{s_1 + s_2}. \tag{1}$$

In addition to being simple, this formula is nice for another reason. Consider the following game: two players start with stacks s_1 and s_2, and each round they flip a coin and the loser pays the winner some fixed number ϵ chips until one player has no more chips. This is just the gambler's ruin problem, with the same solution: player i wins with probability $\frac{s_i}{s_1+s_2}$.

An additional noteworthy conclusion of [13] is that the optimal strategy involves very little randomization: at each state each player only needs to randomize (play an action with probability not equal to 0 or 1) with at most one hand. They also note that there is no general strength-ranking of hands: there exist hands A and B and stack vectors v and w such that A should be jammed and B should be folded at v, but B should be jammed and A should be folded at w. Finally, they prove that the optimal strategy for a single hand of a cash game is almost identical to that of a tournament.

4. INDEPENDENT CHIP MODEL (ICM)

It is not clear which of the conclusions that hold for two players extend to more players; however, one thing that is clear is that chips and money do not necessarily coincide with more than two players (while they do with two players). For example, suppose the chip stacks are 5000, 4900, and 100 with blinds at 50/100. If the player with 100 folds and the player with 5000 jams, should the player A with 4900 be indifferent between taking a 50-50 for all his chips (in this hypothetical example we assume he is sure it will be a 50-50 if he calls)? In a single hand of a cash game he should be indifferent, because the expected payoff of calling and folding are both 0 (ignoring the blinds which are negligible compared to the relevant stack sizes). However this is not the case in a tournament with 50/30/20 payoffs. If he folds, then the short stack will be eliminated very soon and A will expect to come in first and second with probability $\frac{1}{2}$: so his expected payoff is \$40. If he calls, then he will basically guarantee winning the tournament if he wins, but will come in third if he loses. So his expected payoff is \$35. So he will gain \$5 on average by folding.

Unfortunately, there is no obvious way of generalizing equation (1) to come up with a heuristic for performing expected value calculations in tournaments with more than two players; however, the following formula has been proposed and has received widespread acceptance in the poker community for the last several years [15]:

$$\begin{aligned} U_i &= P_1 * \frac{s_i}{S} + P_2 \times \sum_{j \neq i} \left[\frac{s_j}{S} \times \frac{s_i}{S - s_j} \right] \\ &+ P_3 \times \sum_{j \neq i} \sum_{k \neq i, k \neq j} \left[\frac{s_j}{S} \times \frac{s_k}{S - s_j} \times \frac{s_i}{S - s_j - s_k} \right], \end{aligned}$$

where U_i denotes the expected payoff (ignoring the entry fee)

of player i, s_i denotes his stack, $S = \sum_j s_j$, and P_1, P_2, P_3 are the payoffs to the top three finishers. This formula is referred to as the *Independent Chip Model* (ICM), and all of the popular tournament software tools (such as [14]) use ICM to determine expected payoffs.

5. OUR TWO-LEVEL ITERATIVE ALGORITHM FOR EQUILIBRIUM FINDING

We will now present our approach. It consists of iteration at two nested levels. In the "inner loop," we use an extension of fictitious play to determine ϵ-equilibrium strategies for playing a hand at a given state (stack vector) given the values of possible future states. We do this for all states. Then, in the iteration of the "outer loop," we use value iteration to adjust the values of the different states in light of the new payoffs obtained from the inner loop. Then we execute the inner loop again, then the outer loop, and so on. We terminate when no state's payoff changes by more than δ between outer loop iterations.

In the following subsection, we describe the inner loop. It entails generalizing fictitious play to games of incomplete information. In the subsection after that, we will discuss the outer loop in detail.

5.1 Iteration at a given state using fictitious play

Our basic inner-loop algorithm is an extension of fictitious play to multiplayer incomplete-information games (a similar algorithm was used by [1] in the case of two players in a single hand of a cash game). In standard fictitious play, each player plays a best response to the average strategies of his opponents thus far. More formally, each player i applies the following update rule at each time step t :

$$s_{i,t} = \left(1 - \frac{1}{t}\right) s_{i,t-1} + \frac{1}{t} s'_{i,t},$$

where $s'_{i,t}$ is a best response of player i to the profile $s_{-i,t-1}$ of the other players played at time $t-1$ (strategies can be initialized arbitrarily at $t = 0$). This algorithm was originally developed as a simple learning model for repeated games, and was proven to converge to a Nash equilibrium in two-player zero-sum games [5]; however, it is not guaranteed to converge to an equilibrium in two-player general-sum games or games with more than two players.

In extensive-form games the strategy spaces can be exponential in the number of possible private signals. In particular, in Texas hold'em there are 169 strategically distinct preflop hands (see, e.g., [7]) and two jam/fold moves each player can make at each information set. Therefore, the strategy spaces (for any given stack vector) are of size 2^{169}, 2×2^{169}, and 3×2^{169} for the button, small blind, and big blind, respectively (the big blind does not need to act when the other two players fold). To deal efficiently with this exponential blowup, the algorithm works with the extensive form of the game instead of enumerating the strategies. In the extensive form, each player's best response can be computed by simply traversing his information sets one at a time (assuming the payoffs at the other stack vectors are known). The button has 169 information sets, the small blind has 2×169, and the big blind has 3×169. Therefore, the best response (for each player in turn) can be computed efficiently.

While fictitious play is not guaranteed to converge in our setting, we fortunately have the following result:

THEOREM 1. *[5] Under fictitious play, if the empirical distributions over each player's choices converge, the strategy profile corresponding to the product of these distributions is a Nash equilibrium.*

5.2 Tying the states together: solving the tournament by value iteration

As stated earlier, poker tournaments are stochastic games in which each state corresponds to a vector of stack sizes. In particular, we analyze the tournament in which there are 13500 total chips, and blinds are SB = 300, BB = 600. These correspond to common endgame parameters of sit-and-go tournaments at Pokerstars.com, the most popular online poker site.[2] Each state in our game G is defined by a triple (x_1, x_2, x_3), where x_1 denotes the stack of the button, x_2 denotes the stack of the small blind, x_3 denotes the stack of the big blind, $\sum_i x_i = 13500$, and all x_i are multiples of 300. When one of the x_i becomes zero, that player is eliminated, and we can already solve the remaining game with the prior techniques because there are at most two players left. So we can treat these states as terminal states and substitute the corresponding payoffs to players when we arrive in such states. G has 946 non-terminal states.[3]

Our algorithm for solving the game is the following. First we initialize the assignment V^0 of payoffs to each player at each game state using ICM, described above.[4] Then, suppose we start at some game state x. If we assume that the payoffs of V_0 at all states are the actual values of those game states to the players, then the whole stochastic game just becomes a standard game in which every transition from x leads to a terminal payoff. So we can run the fictitious play algorithm described in the previous section at state x until it (hopefully) converges. Supposing it does converge to an approximate equilibrium y, each player will now have some new expected payoff at state x if profile y is played that might differ from the payoff of V_0. If we do this for all 946 nonterminal game states, we come up with a new assignment V_1 of payoffs to each player at each game state. Now

[2]Actually tournaments on Pokerstars generally have the players post an additional ante of 50 at the beginning of every hand at this blind level, but as in the prior work on jam/fold strategies [13], we ignore these extra antes.

[3]In order to guarantee that we always remain in a valid state of G, we must resolve a slight ambiguity involving split pots. For example, suppose $s = (12300, 600, 600)$, the button jams, the small blind folds, the big blind jams, and the button and big blind end up tying after the community cards are dealt. Then the outcome is that both the button and big blind gain 150 chips and the small blind loses 300 chips. This would take us out of a valid state of our game since the button and big blind's stacks are no longer multiples of 300. Our solution is to use a tiebreaking rule for this situation and state that instead of giving 150 to both the button and big blind, we will give 300 to the button and 0 to the big blind with probability 0.5, and give 300 to the big blind and 0 to the button with probability 0.5.

[4]We recently became aware of a website [12] that claims to use a similar inner loop algorithm to ours on tournaments. Unlike in our work, they make the significantly simplifying assumption of ICM payoffs at all states (we only use ICM for initialization and thus do not have any such assumptions). Furthermore, they ignore certain card removal effects and use restricted strategy spaces.

suppose we repeat this process and that hypothetically the payoffs remain the same between iterations k and $k+1$ (all the payoffs in V_k and V_{k+1} are equal). This would suggest that the strategies computed by fictitious play at iteration k (assuming they converge) are close to an equilibrium of the full game G.

So our overall algorithm for computing an approximate Nash equilibrium of the tournament is as follows. First, fix some $\epsilon, \delta > 0$. Initialize the V_0 to ICM, and run fictitious play using V_0 as payoffs until an ϵ-equilibrium is reached (a strategy profile in which no player can gain more than ϵ in expectation by deviating). Then use the payoffs V_1 which result from the computed strategy profile and repeat. The algorithm halts when it reaches an iteration k such that each payoff of V_k differs from the corresponding payoff of V_{k-1} by less than δ.

Algorithm 5.1: COMPUTEEQUILIBRIUM(ϵ, δ)

$V_0 = ICM$
$DiffOuter = \infty$
$i = 0$
while $(DiffOuter > \delta)$
do
- $i = i + 1$
- $Regret = \infty$
- $Initialize(strat)$
- **while** $(Regret > \epsilon)$
  - **do**
    - $strat = FictPlay(strat)$
    - $Regret = MaxRegret(strat)$
- $V_i = getNewValues(V_{i-1}, strat)$
- $DiffOuter = MaxDev(V_i, V_{i-1})$

return $(strat)$

6. IMPLEMENTATION DISCUSSION

6.1 11-card rollout

The main computational challenge of this project was precomputing the probabilities of all of the events that occur when all three players are all-in preflop (e.g., player 1 wins, players 2 and three tie for second): there are 13 such probabilities for each set of hole cards. In order to do this, we had to perform an 11-card rollout — essentially iterating over all possible hole cards for the players (6 total) and all possible sets of community cards (5 total). The straightforward approach of iterating over all

$$\binom{52}{2}\binom{50}{2}\binom{48}{2}\binom{46}{5} = 2.51 \times 10^{15}$$

possibilities would take too long, and we were forced to find ways of exploiting suit symmetries to hopefully reduce the running time.

6.1.1 Exploiting suit symmetries

Gilpin, Sandholm and Sørensen [10] faced similar issues when they performed a nine-card rollout for two players. Unfortunately their techniques do not extend directly to three players, and we were forced to come up with a new method of exploiting symmetries which works as follows. First, fix an order of the players a, b, c and number the cards from 0 to 51 (0 is 2♣, 1 is 2♢, etc.). Also order the suits from 0 to 3: (♣, ♢, ♡, ♠). For each player, fix an ordering of his hole cards: $a_1 < a_2$, $b_1 < b_2$, and $c_1 < c_2$. We restrict a_1 to be of suit 0, then proceed as follows in the order a_2, b_1, b_2, c_1, c_2. Each of these cards can either be a suit of one of its predecessors, or can be 1 more than the maximum suit of one of its predecessors. For example, a_2 can either have suit 0 or 1, since its only predecessor a_1 had suit 0. Then every set of hole cards for the three players is equivalent (up to a permutation of players and suits) to a set of cards meeting the above requirements; thus we only need to iterate over this smaller set of hole cards in the rollout.

We will now give an example that demonstrates how to transform a particular hand to the canonical form discussed in the preceding paragraph. Suppose the three sets of hole cards are as follows: $(7♠, K♡), (A♢, A♣), (6♢, J♠)$. Since the 6♢ is the lowest of the six cards, we fix the last player to be player 1. Similarly we fix the first player to be player 2 since 7♠ is the lowest card of the remaining players, and the middle player becomes player 3. Since player 1's lowest card must be a club, we must apply a permutation mapping ♢ → ♣. Since 6♢ and $J♠$ are of different suits, we must map ♠ → ♢. So far we have $a_1 = 6♣$, $a_2 = J♢$. Since ♠ has already been mapped to a suit, we set $b_1 = 7♢$. Now $K♡$ does not match any of the suits seen so far: so we must set it to the lowest ranked unseen suit, which is ♡. So our permutation maps ♡ → ♡ and therefore ♣ → ♠. Thus $b_2 = K♡$, $c_1 = A♣$, $c_2 = A♠$.

6.1.2 Indexing in the rollout

The previous section discussed how we were able to reduce the number of possible hole card combinations we needed to iterate over. Once we fixed a set of six hole cards for the players, we now had to iterate over all possible sets of five community cards. Let us denote the hole cards as in the previous section and call the community cards $t_1, \ldots, t_5$. Without loss of generality, we can assume that $t_1 < t_2 < \ldots < t_5$: this reduces the number of community card possibilities we need to iterate over by a factor of $5! = 120$. However, while the authors of [10] were able to come up with some clever ways of further reducing the number of community card possibilities with two players, this proved to be more difficult with three players and we were not able to do so. So for each set of hole cards, we were forced to iterate over $\binom{46}{5} = 1370754$ combinations of community cards.

Given a fixed set of hole cards and community cards, our strategy for computing the desired probabilities was the following. First, we construct the seven-card hand for the first player $(a_1, a_2, t_1, t_2, t_3, t_4, t_5)$, and do the same for the other two players. We next compute the ranking of this hand, which denotes the value of the best possible five-card hand our of these seven cards. Since performing all of these calculations at run time would make the rollout take too long, we precomputed all of these rankings of seven card hands and output them to a file, which we read into an array before we started the seven card rollout. The straightforward method of doing this would require an array with $52^7 = 10^{12}$ elements. Such a large array would slow down the rollout too much, so we were forced to find a way to make it smaller. Fortunately, we were able to apply the same indexing technique used by Gilpin, Sandholm and Sørensen [10, 3]. The *colexicographical index* of a set

of integers $x = \{x_1 \ldots, x_k\} \subset \{0, \ldots, n-1\}$ with $x_i < x_j$ whenever $i < j$ is

$$colex(x) = \sum_{i=1}^{k} \binom{x_i}{i}.$$

This index has the important property that for a given n, each of the $\binom{n}{k}$ sets of size k has a distinct colexicographical index. Furthermore, these indices are compactly encoded as the integers from 0 to $\binom{n}{k} - 1$. Since the ranking of a hand is not affected by the order of the seven cards, we can assume without loss of generality that the cards are given in increasing order. So if we index each seven-card hand by the index obtained after permuting the cards so they are in increasing order, we only require an array of size $\binom{52}{7} = 133784650$. This results in a reduction of memory by a factor of 7685, which proved to be crucial given the number of iterations we had to perform.

With these techniques, the computation of determining the 11-card rollout took a month using 16 processors. The output includes for each combination of three pairs of hole cards the probability distribution over rollout outcomes (who wins, who comes second, and who comes third, treating outcomes with ties counting as separate outcomes). We then stored that database, which is used heavily in our equilibrium-finding algorithm discussed above.

6.2 Indexing within the equilibrium finding

Another implementation issue of note arose when we needed to access the rollout probabilities within our equilibrium computation. First, notice that each player is playing a strategy that only depends on his hand ranking and not the specific cards per se. That is, each player treats $Q\heartsuit 8\spadesuit$ and $Q\diamondsuit 8\clubsuit$ the same strategically, even though the actual cards are different (there are 169 strategically distinct hands). The straightforward method of reading in the output of the 11-card rollout would involve using an array with $52^6 \times 13 = 2.57 \times 10^{11}$ entries (13 outcomes for each set of 6 hole cards). However, for the purposes of our algorithm, we can combine all of the results in which a player has two different but strategically equivalent hands together into a single entry of the array. Thus we can collapse this array into one of size $169^3 \times 13$ which has about 63 million entries. However, even after this reduction the time needed to access elements of this array made the running time too long (since the accesses were nested inside several loops in each iteration of the inner loop).

We observed that we could obtain a further reduction in memory by fixing a permutation of the hand rankings. That is, suppose the three hand rankings are a, b, and c. The array described in the previous paragraph would have up to six different entries corresponding to different permutations of these rankings. However, suppose we fix an ordering of the hand rankings and suppose $a \leq b \leq c$ using this ordering (note that there can be equalities: for example, two players can be dealt pocket aces). Then our new array would just have an element corresponding to the triple (a, b, c) and not all of the other permutations. However, we cannot use the indexing scheme of the previous section because of the possibility of equalities between hand rankings. Fortunately, Troels Sørenson suggested to us the following indexing scheme, which is able to deal with multisets of this form. Now we define the *multiset-colexicographical index* of a set of integers $x = \{x_1 \ldots, x_k\} \subset \{0, \ldots, n-1\}$ with $x_i \leq x_j$ whenever $i < j$ to be

$$multi - colex(x) = \sum_{i=1}^{k} \binom{x_i + i - 1}{i}.$$

Using this indexing scheme, we only require an array of size 881805×13, which has about 11 million entries. This is about a factor of 5.5 further reduction and proved to be enough for our algorithm to run sufficiently fast.

We also found it useful to map each stack vector $s = (s_1, s_2, s_3)$ to a unique index using the following formula, where we assume $0 < s_i \leq n$ and s_i is an integer for each i (we divided the stacks by 300 before applying this):

$$\begin{aligned} stack - colex(s) &= (s_2 - 1) + \sum_{i=1}^{s_1 - 1} \sum_{j=1}^{n-i-1} 1 \\ &= -0.5 s_1^2 + s_1(n - 0.5) + s_2 - n. \end{aligned}$$

7. RESULTS FROM OUR EQUILIBRIUM COMPUTATION

We were able to compute an approximate equilibrium for the tournament with $\epsilon = \$0.001$ and $\delta = 0.05$. (We started running value iterations with $\epsilon = \$0.01$ and decreased it over time.) This required 21 iterations of our outer loop (value iteration), each lasting several hours. Within each value iteration, fictitious play usually converged in several hundred iterations for $\epsilon = \$0.001$ and in less than 100 iterations for $\epsilon = \$0.01$. We also computed an approximate equilibrium in a single hand of a cash game for the same blind and stack parameters, where each chip represents \$1. In this case we compute an ϵ-equilibrium with $\epsilon = \$0.01$.

Although we do not prove that our algorithm can only converge to an equilibrium in the tournament setting, in our follow-up work we have devised a procedure to determine precisely how much a player can gain by deviating from our computed strategy profile [6]. The conclusion is that for any starting state of the tournament, no player can gain more than \$0.0488 by deviating. This represents less than 0.1% of the highest possible payoff and 0.05% of the total prize pool of the tournament.

(In that follow-up paper we also argue that the algorithm in this paper may converge to a strategy profile that is not an equilibrium. In that paper we proceed to present three new algorithms, each of which has the property that if the algorithm converges, it converges to an equilibrium.)

7.1 Assessing the Independent Chip Model

After we computed the approximate equilibrium strategies in the tournament, we compared the payoffs for each player at each stack vector (when our new equilibrium strategy is played) to ICM predictions. Our findings are listed in Table 1. The left column denotes the absolute value of the difference between the ICM prediction and our final result in dollars (where the total prize pool is \$100). Since there are 946 possible stack scenarios and three players, there are 2838 total deviations. The average deviation was \$0.3703, and the range was from $\$4.42 \times 10^{-5}$ to \$2.99.

The largest deviation of \$2.99 occurs at the following stacks: $s_1 = 300, s_2 = 12300, s_3 = 900$ (for button, small blind, big blind). ICM predicts that the prize winnings of the big blind are \$28.848, while we obtain \$25.856. ICM also undervalues the button's prize equity by \$2.49 (\$22.96 vs.

Table 1: Deviations between ICM and our payoffs

Range	Frequency
0-0.1	510
0.1-0.2	460
0.2-0.3	368
0.3-0.4	363
0.4-0.5	403
0.5-1	626
1-3	108

$25.45), and undervalues the small blind's equity by $0.50 ($48.19 vs. $48.69) at these stacks. The reason for this large deviation is that ICM thinks that the big blind is three times as likely to win the tournament as the button (because his stack is three times as large); however, this does not take into account the fact that the big blind must post $\frac{2}{3}$ of his stack as blinds the next hand. If the button folds on the next hand and the small blind jams, the big blind will be getting 6–1 odds to call. If he calls and loses then he will finish in 3rd, while if he wins then he will almost ensure finishing in second. On the other hand, if he folds then both he and the button will be all-in on the next hand, and it will be a coin-flip as to who will finish in second and third. Thus, the prize equities of the button and big blind should actually be very close to each other (as our results confirm).

7.2 Comparing tournament strategies and single-hand strategies

In this section we compare the strategies we computed for tournaments with the strategies we computed for a single hand. Interestingly, the conclusions are very different from those resulting from the same comparison in the two-player game [13]. In the multiplayer setting, there is a big difference between the tournament case and the single-hand case, while in the two-player setting there was not.

Tables 5–10 give our computed approximate equilibrium strategies for all players in a tournament with even chip stacks $s_1 = s_2 = s_3 = 4500$ and blinds at $SB = 300$, $BB = 600$, while tables 11–16 give the strategies for a single hand of a cash game with the same parameters. Each square in the tables represents one of the 169 distinct starting hands, with suited hands being in the upper right and unsuited hands in the lower left. The numbers in each square denote the probability the player should jam with that hand, rounded to the nearest 1%. A 'P' (for 'push') means that the probability of jamming is 100%, and 'F' (for 'fold') means the probability of jamming is 0%. So for example, in Table 7 the small blind should jam 93 suited with probability 0.23, and should always fold 93 unsuited.

Table 2 gives the total probabilities of the different outcomes when both players use their computed strategies in a tournament and a single hand. The chart suggests that the equilibrium strategies are fairly similar in a tournament and single hand for the first player to enter the pot, but that players should be much more aggressive in a single hand when someone else has already jammed. The difference is most significant in the final situation: big blind jams with probability 0.197 in a single hand after both of the other players jam, but only probability 0.021 in a tournament (a factor of 9.4 difference). In a tournament, the big blind only calls with the four highest pocket pairs and AKs ('s' denotes 'suited,' 'o' denotes 'offsuit'). In a single hand, his range includes hands like 33, A9o, K9s, QJo, T8s, and 87s.

Table 2: Differences between tournament and single hand strategy with equal stacks.

Situation	Tournament prob.	Single hand prob.
Button jam	0.376	0.355
SB jam after jam	0.088	0.234
SB jam after fold	0.652	0.633
BB jam after jam/fold	0.145	0.308
BB jam after fold/jam	0.273	0.466
BB jam after jam/jam	0.021	0.197

Interestingly, despite the fact that the small blind jams slightly more often following a fold in the tournament than single hand, the big blind jams following fold/jam almost twice as often in a single hand as in a tournament. Hands that the small blind jams following a fold in a tournament but not a single hand include T3s, 94s, 52s, 96o, 65o. The small blind also jams some hands in a single hand that he folds in a tournament such as Q5o and Q4o. Despite the fact that the small blind is jamming less often following a fold in the single hand, the big blind still calls with dozens of marginal hands that he would fold in a tournament, including 33, 22, A2o, K5s–K2s, K8o–K2o, Q8s–Q3s,Q9o–Q7o, J9s–J7s, JTo–J8o, T9s–T8s, T9o, and 98s.

7.3 Nonexistence of a fixed ranking of hands

Two player results [13] show that there is no single ranking of the hands (i.e., there exist two hands h and h' such that at one stack vector, h should be jammed and not h', but at another stack vector, h' should be jammed and not h). In particular, when SB = 1800, BB = 6200 the small blind should fold 43s and jam J2o, but when SB = 3600 and BB = 4400 the small blind should jam 43s and fold J2o. We obtain similar results with three players. For example, we observe such a phenomenon in the small blind's strategy given a fold between the cases of equal stacks (Table 7) and the stacks: BUT = 3300, SB = 1500, BB = 8700 (Table 3). In the latter, the small blind jams with Q2o–Q5o, J5o–J6o but folds T2s–T4s, 94s–95s, 96o–97o, 84s–85s, 86o–87o, 74s–75s, 76o, 63s–65s, 65o,52s–54s, 43s (all of which mark deviations from the first case).

Table 3: Small blind strategy after button folds with stacks (3300, 1500, 8700).

	A	K	Q	J	T	9	8	7	6	5	4	3	2
A	P	P	P	P	P	P	P	P	P	P	P	P	P
K	P	P	P	P	P	P	P	P	P	P	P	P	P
Q	P	P	P	P	P	P	P	P	P	P	P	P	P
J	P	P	P	P	P	P	P	P	P	P	99	97	91
T	P	P	P	P	P	P	P	P	99	90	F	F	F
9	P	P	P	P	P	P	P	P	95	F	F	F	F
8	P	P	P	P	P	97	P	97	90	F	F	F	F
7	P	P	P	P	95	21	F	P	85	F	F	F	F
6	P	P	P	96	F	F	F	F	P	F	F	F	F
5	P	P	P	91	F	F	F	F	F	P	F	F	F
4	P	P	P	F	F	F	F	F	F	F	P	F	F
3	P	P	97	F	F	F	F	F	F	F	F	P	F
2	P	P	93	F	F	F	F	F	F	F	F	F	P

It is also interesting to compare our results to the Karlson-Sklansky (K-S) hand ranking system; these rankings are given in Table 4, where 0 denotes the best hand and 168 denotes the worst hand. This system has often been proposed as a good heuristic for evaluating hand strength, and many software tools use hand ranges based on K-S rankings. However, our results for the even stack case show that equilibrium strategies might drastically contradict K-S rankings. For example, in the even stack case after the button has folded (Table 7) the small blind should fold Q5o (K-S

ranking 84), but jam with 52s (K-S ranking 155). Interestingly, it seems that equilibrium strategy actually agrees pretty closely with K-S rankings for players acting after another player has already jammed. For example, strategy for the big blind contradicts K-S rankings only on a few hands.

Table 4: Karlson-Sklansky hand rankings

	A	K	Q	J	T	9	8	7	6	5	4	3	2
A	0	2	6	10	12	17	19	23	27	25	28	32	34
K	4	1	20	24	30	42	46	47	50	53	55	58	59
Q	8	33	3	39	45	52	60	67	69	72	75	77	81
J	13	38	51	5	48	62	71	79	87	89	91	96	99
T	16	44	56	65	7	68	78	88	97	106	107	112	116
9	22	49	64	76	86	9	83	94	104	114	123	127	132
8	26	54	74	85	95	102	11	100	108	118	128	138	141
7	31	57	80	92	103	111	117	14	113	122	133	142	151
6	36	61	82	101	110	119	126	131	15	125	137	146	155
5	35	63	84	105	120	129	136	140	144	18	134	145	154
4	37	66	90	109	124	139	147	150	153	152	21	149	157
3	40	70	93	115	130	143	156	159	161	160	163	29	162
2	43	73	98	121	135	148	158	164	166	165	167	168	41

7.4 Rarity of randomization

As in the two-player case [13], we observe that approximate equilibrium strategy for three players involves little randomization (many of the probabilities are actually very close — but not equal to — 0% or 100% because we halted fictitious play as soon as the ϵ-approximation guarantee was met which did not allow it to fully converge). For example, the tables show that in the case of even stacks, our equilibrium involves randomization on only a few hands. In fact, the small blind's strategy after button jams, and the big blind's strategy following two jams, involve no randomization. In all other stack sizes we make the similar observation that each player only needs to randomize on at most a small number of hands.

8. CONCLUSIONS AND FUTURE WORK

We computed an approximate Nash equilibrium among jam/fold strategies in a no-limit Texas hold'em tournament with three players. Our results show that many of the phenomena which occur in jam/fold equilibrium strategies with two players also occur with three players, such as the nonexistence of a fixed ranking of hands and using very little randomness. However, while equilibrium strategies in a tournament and single hand of a cash game are very similar in the two-player setting, we showed that they differ substantially with three players and that play tends to be much more aggressive in a single hand. We also analyzed a widely accepted heuristic known as the Independent Chip Model and observed that in some cases its predictions differ substantially from the expected payoffs in our approximate equilibrium strategy profile.

Several important questions remain to be answered. First, it would be nice to show that our computed strategies actually constitute an approximate equilibrium in the full tournament when we allow play other than jam/fold (we conjecture that this is the case). This appears much more difficult to do with three players than two players, partially because of the possibility of collusion between players. We also conjecture that colluders cannot hurt a player very much if the player restricts himself to jam/fold strategies, because such strategies reduce the others' strategy spaces drastically. We thus suspect that our strategies might also achieve approximate maximin payoffs for each player. We also suspect that the equilibrium for the three player tournament might be unique (at least with respect to information sets that have nonzero probability of being reached); however, proving this seems difficult.

9. REFERENCES

[1] J. Ankenman and B. Chen. *The Mathematics of Poker*. ConJelCo LLC, 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 *Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI)*, 2003.

[3] B. Bollobás. *Combinatorics*. Cambridge University Press, Cambridge, 1986.

[4] X. Chen and X. Deng. Settling the complexity of 2-player Nash equilibrium. In *Proceedings of the Annual Symposium on Foundations of Computer Science (FOCS)*, 2006.

[5] D. Fudenberg and D. Levine. *The Theory of Learning in Games*. MIT Press, 1998.

[6] S. Ganzfried and T. Sandholm. Algorithms for multiplayer stochastic games of imperfect information with application to three-player no-limit Texas hold'em tournaments. Draft.

[7] A. Gilpin and T. Sandholm. A competitive Texas Hold'em poker player via automated abstraction and real-time equilibrium computation. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, 2006.

[8] A. Gilpin and T. Sandholm. Better automated abstraction techniques for imperfect information games, with application to Texas Hold'em poker. In *International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, 2007.

[9] A. Gilpin and T. Sandholm. Lossless abstraction of imperfect information games. *Journal of the ACM*, 54(5), 2007.

[10] A. Gilpin, T. Sandholm, and T. B. Sørensen. Potential-aware automated abstraction of sequential games, and holistic equilibrium analysis of Texas Hold'em poker. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, 2007.

[11] A. Gilpin, T. Sandholm, and T. B. Sørensen. A heads-up no-limit texas hold'em poker player: Discretized betting models and automatically generated equilibrium-finding programs. In *International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, 2008.

[12] Hold'emResources.net. www.holdemresources.net/hr/sngs/icmcalculator.html.

[13] P. B. Miltersen and T. B. Sørensen. A near-optimal strategy for a heads-up no-limit Texas Hold'em poker tournament. In *International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, 2007.

[14] Sit-n-Go Power Tools, 2005. http://www.sitngo-analyzer.com/.

[15] Two Plus Two Forums. Single table tournament forum FAQ, 2007. http://forumserver.twoplustwo.com/.

Tournament strategy with equal stacks

(suited hands in upper right, unsuited in lower left)

Table 5: Button

	A	K	Q	J	T	9	8	7	6	5	4	3	2
A	P	P	P	P	P	P	P	P	P	P	P	P	P
K	P	P	P	P	P	P	99	99	99	99	99	97	25
Q	P	P	P	P	99	99	99	96	95	31	8	1	1
J	P	P	99	P	99	99	99	94	F	F	F	F	F
T	P	P	99	99	P	99	99	94	F	F	F	F	F
9	P	99	95	93	94	P	99	95	6	F	F	F	F
8	P	13	1	F	F	F	P	96	17	F	F	F	F
7	P	5	F	F	F	F	F	P	82	F	F	F	F
6	99	2	F	F	F	F	F	F	P	F	F	F	F
5	99	1	F	F	F	F	F	F	F	P	F	F	F
4	99	1	F	F	F	F	F	F	F	F	P	F	F
3	99	F	F	F	F	F	F	F	F	F	F	99	F
2	99	F	F	F	F	F	F	F	F	F	F	F	99

Table 6: Small blind after button jams

	A	K	Q	J	T	9	8	7	6	5	4	3	2
A	P	P	P	P	99	98	F	F	F	F	F	F	F
K	P	P	F	F	F	F	F	F	F	F	F	F	F
Q	P	F	P	F	F	F	F	F	F	F	F	F	F
J	P	F	F	P	F	F	F	F	F	F	F	F	F
T	99	F	F	F	P	F	F	F	F	F	F	F	F
9	F	F	F	F	F	P	F	F	F	F	F	F	F
8	F	F	F	F	F	F	99	F	F	F	F	F	F
7	F	F	F	F	F	F	F	98	F	F	F	F	F
6	F	F	F	F	F	F	F	F	F	F	F	F	F
5	F	F	F	F	F	F	F	F	F	F	F	F	F
4	F	F	F	F	F	F	F	F	F	F	F	F	F
3	F	F	F	F	F	F	F	F	F	F	F	F	F
2	F	F	F	F	F	F	F	F	F	F	F	F	F

Table 7: Small blind after button folds

	A	K	Q	J	T	9	8	7	6	5	4	3	2
A	P	P	P	P	P	P	P	P	P	P	P	P	P
K	P	P	P	P	P	P	P	P	P	P	P	P	P
Q	P	P	P	P	P	P	P	P	99	99	99	99	99
J	P	P	P	P	P	P	P	99	99	99	99	99	94
T	P	P	P	P	P	P	P	99	99	99	99	93	88
9	P	P	P	99	99	P	P	99	99	99	90	23	2
8	P	P	99	99	99	99	P	99	99	99	99	2	2
7	P	P	91	91	99	99	99	P	99	99	99	3	2
6	P	P	90	3	3	87	94	99	P	99	99	92	2
5	P	99	3	2	1	2	2	3	91	P	99	99	85
4	P	99	3	2	1	1	1	1	2	3	P	93	2
3	P	99	2	1	F	F	F	F	F	1	1	P	2
2	P	99	2	1	F	F	F	F	F	F	F	F	P

Table 8: Big blind after jam/fold

	A	K	Q	J	T	9	8	7	6	5	4	3	2
A	P	P	P	P	P	P	99	99	89	87	F	F	F
K	P	P	99	99	89	F	F	F	F	F	F	F	F
Q	P	99	P	F	F	F	F	F	F	F	F	F	F
J	P	F	F	P	F	F	F	F	F	F	F	F	F
T	P	F	F	F	P	F	F	F	F	F	F	F	F
9	99	F	F	F	F	P	F	F	F	F	F	F	F
8	99	F	F	F	F	F	P	F	F	F	F	F	F
7	F	F	F	F	F	F	F	P	F	F	F	F	F
6	F	F	F	F	F	F	F	F	P	F	F	F	F
5	F	F	F	F	F	F	F	F	F	99	F	F	F
4	F	F	F	F	F	F	F	F	F	F	15	F	F
3	F	F	F	F	F	F	F	F	F	F	F	F	F
2	F	F	F	F	F	F	F	F	F	F	F	F	F

Table 9: Big blind after fold/jam

	A	K	Q	J	T	9	8	7	6	5	4	3	2
A	P	P	P	P	P	P	P	P	P	P	P	P	99
K	P	P	P	P	P	P	99	99	87	4	F	F	F
Q	P	P	P	99	99	98	3	F	F	F	F	F	F
J	P	P	99	P	98	1	F	F	F	F	F	F	F
T	P	P	91	F	P	F	F	F	F	F	F	F	F
9	P	99	F	F	F	P	F	F	F	F	F	F	F
8	P	6	F	F	F	F	P	F	F	F	F	F	F
7	P	2	F	F	F	F	F	P	F	F	F	F	F
6	99	F	F	F	F	F	F	F	P	F	F	F	F
5	99	F	F	F	F	F	F	F	F	P	F	F	F
4	99	F	F	F	F	F	F	F	F	F	P	F	F
3	99	F	F	F	F	F	F	F	F	F	F	3	F
2	5	F	F	F	F	F	F	F	F	F	F	F	F

Table 10: Big blind after jam/jam

	A	K	Q	J	T	9	8	7	6	5	4	3	2
A	P	96	1	F	F	F	F	F	F	F	F	F	F
K	2	P	F	F	F	F	F	F	F	F	F	F	F
Q	F	F	P	F	F	F	F	F	F	F	F	F	F
J	F	F	F	P	F	F	F	F	F	F	F	F	F
T	F	F	F	F	1	F	F	F	F	F	F	F	F
9	F	F	F	F	F	P	F	F	F	F	F	F	F
8	F	F	F	F	F	F	P	F	F	F	F	F	F
7	F	F	F	F	F	F	F	P	F	F	F	F	F
6	F	F	F	F	F	F	F	F	P	F	F	F	F
5	F	F	F	F	F	F	F	F	F	P	F	F	F
4	F	F	F	F	F	F	F	F	F	F	P	F	F
3	F	F	F	F	F	F	F	F	F	F	F	P	F
2	F	F	F	F	F	F	F	F	F	F	F	F	P

Single hand strategy with equal stacks

(suited hands in upper right, unsuited in lower left)

Table 11: Button

	A	K	Q	J	T	9	8	7	6	5	4	3	2
A	P	P	P	P	P	P	P	P	P	P	P	P	P
K	P	P	P	P	P	P	P	P	P	P	P	98	F
Q	P	P	P	P	P	P	P	96	94	F	F	F	F
J	P	P	P	P	P	P	98	95	F	F	F	F	F
T	P	P	P	P	P	P	98	96	F	F	F	F	F
9	P	P	F	F	20	P	99	97	F	F	F	F	F
8	P	2	F	F	F	F	P	98	95	F	F	F	F
7	P	F	F	F	F	F	F	P	97	F	F	F	F
6	P	F	F	F	F	F	F	F	P	94	F	F	F
5	P	F	F	F	F	F	F	F	F	P	F	F	F
4	P	F	F	F	F	F	F	F	F	F	P	F	F
3	P	F	F	F	F	F	F	F	F	F	F	P	F
2	P	F	F	F	F	F	F	F	F	F	F	F	P

Table 12: Small blind after button jams

	A	K	Q	J	T	9	8	7	6	5	4	3	2
A	P	P	P	P	P	P	P	P	P	P	P	P	P
K	P	P	P	P	P	P	1	1	1	F	F	F	F
Q	P	P	P	P	P	1	F	F	F	F	F	F	F
J	P	P	91	P	P	1	F	F	F	F	F	F	F
T	P	P	1	F	P	1	F	F	F	F	F	F	F
9	P	1	F	F	F	P	F	F	F	F	F	F	F
8	P	F	F	F	F	F	P	F	F	F	F	F	F
7	P	F	F	F	F	F	F	P	F	F	F	F	F
6	P	F	F	F	F	F	F	F	P	F	F	F	F
5	P	F	F	F	F	F	F	F	F	P	F	F	F
4	1	F	F	F	F	F	F	F	F	F	P	F	F
3	1	F	F	F	F	F	F	F	F	F	F	P	F
2	1	F	F	F	F	F	F	F	F	F	F	F	P

Table 13: Small blind after button folds

	A	K	Q	J	T	9	8	7	6	5	4	3	2
A	P	P	P	P	P	P	P	P	P	P	P	P	P
K	P	P	P	P	P	P	P	P	P	P	P	P	P
Q	P	P	P	P	P	P	P	P	P	P	P	P	P
J	P	P	P	P	P	P	P	P	P	P	99	99	99
T	P	P	P	P	P	P	P	P	99	99	99	17	F
9	P	P	P	P	P	P	P	P	99	99	F	F	F
8	P	P	P	P	P	99	P	P	99	99	98	F	F
7	P	P	P	P	99	99	99	P	99	99	98	F	F
6	P	P	P	F	F	F	8	98	P	99	99	11	F
5	P	P	P	F	F	F	F	F	F	P	99	98	F
4	P	P	P	F	F	F	F	F	F	F	P	98	F
3	P	P	4	F	F	F	F	F	F	F	F	P	F
2	P	P	F	F	F	F	F	F	F	F	F	F	P

Table 14: Big blind after jam/fold

	A	K	Q	J	T	9	8	7	6	5	4	3	2
A	P	P	P	P	P	P	P	P	P	P	P	P	P
K	P	P	P	P	P	P	P	P	P	92	1	1	1
Q	P	P	P	P	P	P	89	1	1	P	F	F	F
J	P	P	P	P	P	95	1	F	F	F	F	F	F
T	P	P	P	48	P	95	1	F	F	F	F	F	F
9	P	P	1	F	F	P	F	F	F	F	F	F	F
8	P	1	F	F	F	F	P	F	F	F	F	F	F
7	P	1	F	F	F	F	F	P	F	F	F	F	F
6	P	1	F	F	F	F	F	F	P	F	F	F	F
5	P	1	F	F	F	F	F	F	F	P	F	F	F
4	P	F	F	F	F	F	F	F	F	F	P	F	F
3	P	F	F	F	F	F	F	F	F	F	F	P	F
2	P	F	F	F	F	F	F	F	F	F	F	F	P

Table 15: Big blind after fold/jam

	A	K	Q	J	T	9	8	7	6	5	4	3	2
A	P	P	P	P	P	P	P	P	P	P	P	P	P
K	P	P	P	P	P	P	P	P	P	P	P	P	P
Q	P	P	P	P	P	P	P	P	P	P	99	96	1
J	P	P	P	P	P	P	P	98	1	F	F	F	F
T	P	P	P	P	P	P	P	5	F	F	F	F	F
9	P	P	P	P	97	P	99	1	F	F	F	F	F
8	P	P	P	93	F	F	P	F	F	F	F	F	F
7	P	P	97	F	F	F	F	P	F	F	F	F	F
6	P	P	1	F	F	F	F	F	P	F	F	F	F
5	P	P	F	F	F	F	F	F	F	P	F	F	F
4	P	P	F	F	F	F	F	F	F	F	P	F	F
3	P	P	F	F	F	F	F	F	F	F	F	P	F
2	P	99	F	F	F	F	F	F	F	F	F	F	P

Table 16: Big blind after jam/jam

	A	K	Q	J	T	9	8	7	6	5	4	3	2
A	P	P	P	P	P	P	P	P	P	P	5	3	2
K	P	P	P	P	P	P	2	1	1	1	F	F	F
Q	P	P	P	P	P	P	2	F	F	F	F	F	F
J	P	P	P	P	P	P	3	F	F	F	F	F	F
T	P	8	2	2	P	P	P	F	F	F	F	F	F
9	P	1	F	F	F	P	P	2	F	F	F	F	F
8	4	F	F	F	F	F	P	P	F	F	F	F	F
7	2	F	F	F	F	F	F	P	7	F	F	F	F
6	1	F	F	F	F	F	F	F	P	F	F	F	F
5	1	F	F	F	F	F	F	F	F	P	F	F	F
4	1	F	F	F	F	F	F	F	F	F	P	F	F
3	1	F	F	F	F	F	F	F	F	F	F	P	F
2	F	F	F	F	F	F	F	F	F	F	F	F	2