Automated Action Abstraction of Imperfect Information Extensive-Form Games

John Hawkin and **Robert Holte** and **Duane Szafron**
{hawkin, holte}@cs.ualberta.ca, dszafron@ualberta.ca
Department of Computing Science
University of Alberta
Edmonton, AB, Canada T6G2E8

Abstract

Multi-agent decision problems can often be formulated as extensive-form games. We focus on imperfect information extensive-form games in which one or more actions at many decision points have an associated continuous or many-valued parameter. A stock trading agent, in addition to deciding whether to buy or not, must decide how much to buy. In no-limit poker, in addition to selecting a probability for each action, the agent must decide how much to bet for each betting action. Selecting values for these parameters makes these games extremely large. Two-player no-limit Texas Hold'em poker with stacks of 500 big blinds has approximately 10^{71} states, which is more than 10^{50} times more states than two-player limit Texas Hold'em. The main contribution of this paper is a technique that abstracts a game's action space by selecting one, or a small number, of the many values for each parameter. We show that strategies computed using this new algorithm for no-limit Leduc poker exhibit significant utility gains over ϵ-Nash equilibrium strategies computed with standard, hand-crafted parameter value abstractions.

Introduction

A successful abstraction technique creates a smaller game that retains the important strategic features of the larger game. An algorithm such as CFR (Zinkevich et al. 2008) is then used to compute an ϵ-Nash equilibrium for the smaller game and this is used as an agent in the original game. Recent progress has been made creating algorithms to find ϵ-Nash equilibrium solutions to imperfect information extensive-form games (Zinkevich et al. 2008; Gilpin and Sandholm 2007b). State abstraction of extensive-form games has been considered in detail (Gilpin, Sandholm, and Sorensen 2007; Gilpin and Sandholm 2007a; Waugh et al. 2009a; 2009b). Application of the advances in these two research directions has led to very strong two-player limit Texas Hold'em agents, some capable of beating world class human players.[1]

In games such as two-player no-limit poker, the action space is very large (10^{71} states for stacks of 500 big blinds), and some form of action abstraction must be performed before we can attempt to solve the game. The problem of action abstraction in this domain can be divided into two parts - choosing which bet sizes to remove from the abstraction, and determining how to react to opponent bet sizes that are no longer part of our abstraction. The later is known as the translation problem. Although the problems are not actually independent, current state-of-the art addresses them orthogonally. For example, all game-theory based entries in the annual computer poker competition employ some form of translation. The translation problem has been studied by Schnizlein (Schnizlein 2009; Schnizlein, Bowling, and Szafron 2009), where a fixed action abstraction is used to solve the game, and then actions taken by the opponent are mapped to actions within the chosen action abstraction in a probabilistic way.

Little research has been done on discovering action space abstractions that prune unimportant actions. In games with very large action spaces modern agents often use very simple action abstractions - agents developed to play no-limit Texas Hold'em often only ever bet the size of the pot or all their chips, or in addition a few other amounts dependent on the size of the pot such as fractions or multiples of the pot size (Gilpin, Sandholm, and Sorensen 2008; Schnizlein 2009). A great deal of value can be gained in poker games from choosing appropriate bet sizes - in the small game of half-street Kuhn poker, for example, the betting player has the advantage if they choose their bet size well, but for bet sizes of pot and greater this advantage is reduced to zero. The bettor has lost their advantage by selecting an inferior bet size (Chen and Ankenman 2007).

We introduce a new transformation where the task of choosing the parameter value for a decision agent is assigned to a new agent who has the same utility as the decision agent. The transformed game is a multi-agent game consisting of two teams, where each team has a single decision agent and possibly multiple parameter-setting agents, one for each parameter. We demonstrate the usefulness of this transformation in three ways. First, we show that known Nash equilibrium solutions for small no-limit poker games map to computed Nash equilibrium solutions of the transformed game. Second, we provide empirical evidence that ϵ-Nash equilibrium solutions to the transformed game can be found by applying a new regret-minimizing algorithm based on the CFR algorithm. Finally, we experimentally show that strategies computed using this new algorithm for no-limit Leduc poker exhibit significant utility gains over ϵ-Nash equilib-

[1] http://www.manmachinepoker.com

rium strategies computed with standard, hand-crafted action abstractions. These results suggest that this technique can be used to select better parameter values in very large multi-agent decision problems with parameterized actions, such as full no-limit Texas Hold'em.

Background

An extensive form game consists of:

- N players.
- A set H of histories. Each history h in this set consists of a sequence of valid actions for the game (made by players and chance).
- A set $Z \subseteq H$ of terminal histories. A terminal history represents the end of the game - no more actions allowed.
- A player function P that determines whose turn it is at each point in the game. This can be $P = i$, meaning player i's turn, or $P = c$, where c is chance.
- A function f_c defining the chance probabilities.
- An information partition $\mathbf{I}_i$ for each player i consisting of all of the information sets I_i for that player. An information set is a set of histories that the player cannot distinguish from one another.
- A utility function u_i for each player i assigning a real value to each terminal history. This is the payoff for that player when that terminal history is reached.

Game theory definitions

To play an extensive form game each player must have a strategy. A strategy is a probability distribution over all legal actions available to that player for every possible history. A strategy profile σ is a set of strategies, one for each player, with σ_i being the strategy of player i. $\sigma_i(I, a)$ is the probability that player i will take action a in information set I if they act according to σ. $u_i(\sigma)$ is the utility of strategy profile σ to player i. Given a strategy profile, a best response is a strategy for one player that maximizes that player's expected utility when played against the strategies in that profile. A Nash equilibrium is a strategy profile such that no player can improve their utility by changing their strategy. In a two-player zero-sum game the value of the game for a player is that player's expected utility if both players follow a Nash equilibrium strategy profile. An ϵ-Nash equilibrium is a strategy profile in which no player can improve by more than ϵ by changing strategy.

Previous work on action abstraction

The two main approaches that have been pursued when considering domains with large or continuous action spaces are either to sample from the continuous action space according to some distribution or to create coarse grain abstractions of the action space and apply standard methods for discrete action spaces. Lazaric et al. (2008) tackle domains with continuous action spaces by using an actor critic approach in which the policy of the actor is estimated using sequential Monte Carlo methods. They then use importance sampling on the basis of the values learned by the critic, and they apply resampling to modify the actor's policy. They show results where their algorithm performs better than static solutions such as continuous Q-learning. Hasselt et al. (2007) also designed an actor critic algorithm for continuous action spaces, where they only use the sign of the TD-error to determine the update to the actor instead of using the exact value. Their algorithm performed well in simple tracking and balancing problems.

Madeira et al. (2006) developed very coarse-grained abstractions of the state and action space of large-scale strategy games. The resulting agents performed at an intermediate level on the Battleground simulator.

Dean et al. (1998) create coarse-grained abstractions to compactly represent MDPs for planning problems with large state and action spaces. Their methods work most effectively in domains where few of the actions have much impact on achieving the final goal or minimizing costs.

Little theoretical work exists dealing with agents operating in domains with large or continuous action spaces. Tijs (Tijs 1980) shows that for infinite stage stochastic games with very large action spaces there exist ϵ-equilibrium points for all $\epsilon > 0$. Antos et al. (Antos, Munos, and Szepesvari 2007) provide a rigorous analysis of a fitted Q-iteration algorithm for continuous action space MDPs, deriving what they believe to be the first finite-time bound for value-function based algorithms for continuous state and action problems.

Rules of heads up no-limit poker

In this paper we will use various two-player no-limit poker games as our test bed. In two-player no-limit poker games each player starts with the same number of chips, known as their stack. In the games we study, each player is required to put some number of chips, known as the "ante", in the pot. Next, cards are dealt to each player, at least one of which is dealt face down and kept private. The betting round now begins. Players may *fold*, surrendering the pot to the other player, *call*, matching whatever bet the opponent has made, or *raise*, matching any bet made by the opponent and increasing it. A call when no bets have been made in the round is known as a *check*, and a raise when no previous raise has happened yet is known simply as a *bet*. In this paper we use "bet" to refer to both bets and raises. If no bet is made after both players are given a chance to act, then the betting round ends. If a bet is made then the other player must respond to this action before continuing to the next round. In no-limit poker games the player chooses the bet size, while in limit games the bet sizes are fixed. A bet of all of a player's remaining chips is known as an All-in bet.

Kuhn poker Kuhn poker is a small poker game invented by H. W. Kuhn (1950). It is played with three cards - one Jack (lowest value), one Queen and one King (highest value). Each player receives one card. After the betting round the game ends. One of the games we consider in this paper is half-street Kuhn poker, where the second player is disallowed from betting or raising.

Leduc poker Leduc poker (Waugh, Bard, and Bowling 2009) uses six cards: two Jacks, two Queens and two Kings.

Each player receives one card. After the first betting round a community card, shared by the players, is dealt and the second betting round begins. If neither player folds the winner is the player who can make the best two-card hand.

The Multiplayer betting game transformation

We now define the multiplayer betting game transformation. It can be applied to domains with actions that have associated real or multi-valued parameters. In the case of poker, the action in question is a bet, and the multi-valued parameter is the size of the bet. These are combined to create a large number of possible actions of the form "bet x", where x is an integer between some minimum value and the stack size of the betting player. In the case of a trading agent, each of the actions buy and sell may have a parameter that specifies the amount. For brevity, we will use poker terminology throughout this paper.

At each decision point where betting is legal we remove most of the betting actions, leaving only an All-in action and a new action we refer to as "bet". The new bet action is then followed by another decision point for a new player, who we will refer to as a "bet sizing player". This player chooses between a low bet and a high bet, which we refer to as betting L or betting H, where these are expressed as fractions of a pot sized bet. At each decision point bet sizes are chosen by a different bet sizing player. The low and high bets can be different for each bet sizing player, with the only restrictions being that for each bet sizing player i $L_i < H_i$ and L_i is at least a minimum bet. The new bet sizing players do not see any of the private cards, and each bet sizing player obtains the exact same payoffs as the player for whom they are choosing the bet. So instead of two players, the game has two teams. The decision of whether to bet L or H is private to the player that chose it - no other player in the game observes this action. In this paper we will refer to the two players that have fold, call, bet and All-in actions as player one and player two.

Figure 1(a) shows a decision point for player one in a no-limit game with a 4 chip pot, 6 chips left in each stack and a minimum bet of 2. The corresponding decision point for the multiplayer betting version of this game is in Figure 1(b). Here circles represent player one, squares represent player two, the diamond is a bet sizing player, and we can choose any values of L and H such that $L \geq 0.5$ and $L < H \leq 1.5$.

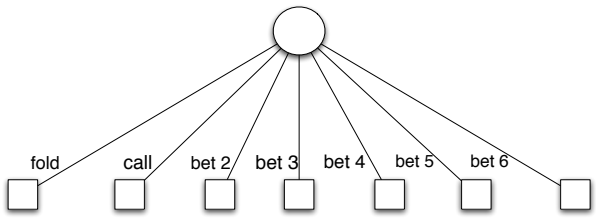
Strategies to play the multiplayer betting game are almost the same size as an abstraction of the full no-limit game that, at each decision point where betting is legal, allows each player to make an All-in bet and one other bet. Strategies in the multiplayer betting game define exactly one extra parameter - $P(H)$ - for each bet sizing player.

We define the effective bet size function

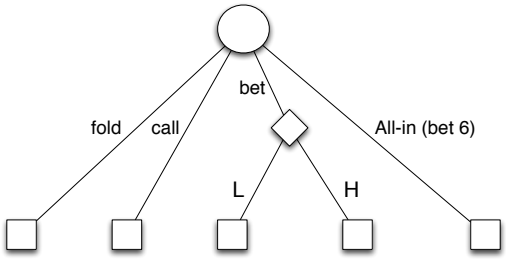
$$B(P(H)) = (1 - P(H))L + P(H)H. \quad (1)$$

We used $P(H) + P(L) = 1$ to eliminate $P(L)$.

Lemma 1. *For any fixed values of L and H and probability $P(H)_i$ of player i betting H, the expected value of the pot size after that bet for all players j such that $j \neq i$ is equal to $p + B(P(H)_i)p$, where p is the pot size before the bet.*



(a) Regular no-limit game



(b) Multiplayer betting game

Figure 1: Decision point in a no-limit game, and the multiplayer betting version of the same decision point.

Proof. After a bet of size L the pot size is $p+pL$, and after a bet of size H the pot size is $p+pH$. So the expected pot size is $P(L)_i(p+pL)+P(H)_i(p+pH) = P(L)_ip+P(H)_ip+P(L)_ipL + P(H)_ipH$. Since $P(L)_i + P(H)_i = 1$, this is equivalent to $p + ((1 - P(H)_i)L + P(H)_iH)\,p$, and using Equation 1 we get $p + B(P(H)_i)p$, as required. □

Theorem 1. *For any given strategy profile σ of a multiplayer betting game $u_i(\sigma)$ where $i = 1$ or $i = 2$ is equivalent to $u_i(\sigma')$ in the abstract no-limit game with the same stack sizes where at each corresponding decision point only bet sizes of $s = B(P(H))$ and All-in are allowed, and $\sigma'_i = \sigma_i$ for $i = 1, 2$.*

Here is a sketch of the proof of Theorem 1. For every terminal history z of the abstract game involving k betting actions there is a corresponding set Z of 2^k terminal histories in the multiplayer betting game where players one and two make the same actions. For example, if $k = 1$ there are two such histories, one where the betting player choses L and one where they choose H. The bet sizes in the abstract game correspond to the probabilities of choosing L or H, so, by Lemma 1, the pot size for history z is equal to the expected pot size for Z. Therefore, the utility of z equals the expected utility for all of Z. Since the probability of z is equal to the combined probability of histories in Z, the utility of a strategy in the abstract game is equal to the utility of the corresponding strategy in the multiplayer game for players one and two.

Theorem 1 allows us to map a given strategy profile σ for a multiplayer betting game to a strategy profile σ' in an abstract poker game, and the strategy has the same value in both games. Each different choice of $P(H)_i$ for each bet sizing player i corresponds to a different abstraction of the

no-limit game, making this choice of abstraction part of a player's strategy in the multiplayer betting game.

Analysis in Kuhn poker

To illustrate the significance of this transformation we derive the Nash equilibrium solution of multiplayer no-limit half-street Kuhn poker. To do this we use analytical formulas previously derived for half-street Kuhn poker with fixed bet sizes that relate Nash equilibrium strategies to the bet size used.

In half-street Kuhn poker there are two non-dominated strategy parameters, $P(b \mid j)$, the probability that the first player bets with the Jack, and $P(c \mid q)$, the probability that the second player calls with the Queen. For any limit version of this game with antes of 0.5 and a fixed real-valued bet size s such that $0 < s < 1$, a strategy profile is a Nash equilibrium if and only if (Chen and Ankenman 2007)

$$P(b \mid j) = \frac{1 - P(c \mid q)}{2} = \frac{s}{s+1} \tag{2}$$

Consider the multiplayer betting version of this game. All bets in half-street Kuhn poker can be represented by an equivalent effective bet size $B(P(H))$. To find a Nash equilibrium for multiplayer betting half-street Kuhn poker we must obey Equation 2 where $s = B(P(H))$, and must also pick $P(H)$ such that the bet sizing player cannot gain by choosing a different value. For this to be true we must have $u_1(B(x)) \leq u_1(s)$, thus $u_1(B(x)) - u_1(s) \leq 0$ for all $0 \leq x \leq 1$. The only situations that affect this difference in utility for a given strategy profile σ are sequences that involve the bet being made and called, so

$$\begin{aligned} u_1(B(x)) - u_1(s) &= \left[\frac{P(b \mid k)}{3}\frac{P(c \mid q)}{2} - \right. \\ &\left.\left(\frac{P(b \mid j)}{3}\left(\frac{P(c \mid k) + P(c \mid q)}{2}\right)\right)\right]\Big(B(x) - s\Big) \\ &= \left[\frac{P(c \mid q)}{6} - \frac{P(b \mid j)}{3}\left(\frac{1 + P(c \mid q)}{2}\right)\right]\Big(B(x) - s\Big) \leq 0. \end{aligned} \tag{3}$$

The first term in the brackets on the right hand side is the probability player one bets, is called and wins, and the second term is the probability player one bets, is called and loses. It is dominated to check or fold the King so we have $P(b \mid k) = 1$ and $P(c \mid k) = 1$. For this to be true for any $B(x)$ we must have

$$P(b \mid j) = \frac{P(c \mid q)}{1 + P(c \mid q)}. \tag{4}$$

To have a Nash equilibrium in the multiplayer betting half-street Kuhn poker game we need to satisfy Equation 4 and Equation 2. So we have

$$\frac{1 - P(c \mid q)}{2} = \frac{P(c \mid q)}{1 + P(c \mid q)}, \tag{5}$$

a quadratic to which the positive solution is

$$P(c \mid q) = \sqrt{2} - 1. \tag{6}$$

Plugging this into Equation 2 and solving for s we get

$$s = \sqrt{2} - 1. \tag{7}$$

So a Nash equilibrium strategy profile in the multiplayer betting half-street Kuhn game must define $P(H)$ such that $B(P(H)) = s = \sqrt{2} - 1$.

Chen and Ankenman also show that the value of half-street Kuhn poker to the betting player as a function of bet size s is given by

$$V(s)_1 = \frac{s(1-s)}{6(1+s)} \tag{8}$$

Figure 2 shows a plot of this function from $s = 0$ to $s = 1$. Each point on this curve represents the game value for a given bet size s. For every point on the x-axis the Nash equilibrium strategy profile may be completely different.

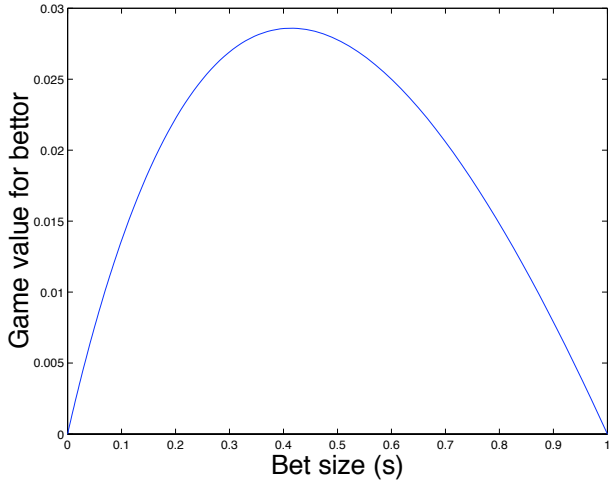


Figure 2: Half-street Kuhn poker game value per bet size.

Chen and Ankenman show that the curve peaks at $s = \sqrt{2} - 1$, thus the Nash equilibrium in no-limit half-street Kuhn poker uses this bet size. This is the same as Equation 7, and thus the choices of $P(b \mid j)$ and $P(c \mid q)$ must be the same in both games. If we can find the Nash equilibrium for multiplayer betting half-street Kuhn poker and map it to a strategy profile for the equivalent abstract no-limit game, then player one will obtain the same game value as in no-limit half-street Kuhn poker. Later in the paper we present our new algorithm that finds improved strategies in multiplayer betting games. First we will show the results of applying this algorithm to a number of poker games.

Computing abstractions for no-limit Kuhn

Applying the algorithm given below, we computed abstractions for three variants of the small poker game Kuhn poker. We then compared the bet sizes we found with existing analytical results (Chen and Ankenman 2007).

For multiplayer betting half-street Kuhn poker our algorithm obtained $B(P(H)) = 0.414$, the bet size for the Nash equilibrium strategy profile of the full no-limit game.

For cases where $H < 0.414$ the algorithm converged with $P(H) = 1$. Figure 2 shows that this is the bet size with the highest game value for the bettor. For choices of $L > 0.414$ we obtained $P(H) = 0$. Next we tested our algorithm on the larger game of no-limit Kuhn poker where each player may make a bet but not raise. The bet sizes used by Nash equilibrium strategies are $s = 0.414$ for player one and $s = 0.509$ for player two (Chen and Ankenman 2007). Again the algorithm converged, with effective bet sizes of $B(P(H)) = 0.414$ and $B(P(H)) = 0.509$ for teams one and two respectively. Finally we looked at the full no-limit Kuhn poker game, where both players can bet and raise indefinitely. As with the previous two games, the bet sizes defining the Nash equilibria are quite unintuitive. The bet sizes are 0.252 for player one's initial bet, 0.542 for player two's bet if player one checks, and the minimum legal raise if player two raises in response to a player one bet (Chen and Ankenman 2007). Using $L = 0.1$ and $H = 0.3$ for player one's bet and $L = 0.4$ and $H = 0.6$ for player two's first bet our algorithm correctly found the 0.252 and 0.542 bet sizes, and for all ranges that we tried for the player two raise, our algorithm set $P(H) = 0$.

Finding bet sizes in larger games

Results in the previous section are from games for which analytical solutions were known, providing a gold standard for evaluating our algorithm. For larger games such as Leduc poker we have neither an analytical expression for the Nash equilibrium, nor can we solve the full no-limit game using current techniques. In order to study such games, our first step is to determine an action abstraction to use as a baseline against which to compare our results. We then find an ϵ-Nash equilibrium using existing techniques, such as the CFR algorithm (Zinkevich et al. 2008), and obtain tight bounds on the game value. Next we create two multiplayer betting games by making one player use the baseline action abstraction and assigning bet sizing players to determine the bets for the other player. We then apply our algorithm to these games, and we use the results to create abstractions by assigning the bet size given by $B(P(H)_i)$ for every bet sizing player i. We create abstractions with large antes (keeping the ratio of the antes to the stack size constant) and round $B(P(H)_i)$ to the nearest chip. We can then get best responses values within these new abstractions, which we can use to determine the game values of each abstraction relative to the others.

Results on Leduc poker

A standard betting abstraction for no-limit games is to allow fold, call, pot bet, and All-in bet actions everywhere (Gilpin, Sandholm, and Sorensen 2008). We used this as our baseline abstraction for Leduc poker. As described above we created two multiplayer betting games, each time replacing the pot bets with bet sizing players with $L = 0.8$ and $H = 1.5$ (but leaving the All-in actions in the games). We allowed a maximum of two raises per round for both the baseline Leduc game and the multiplayer betting games. All games were played using antes, with stacks of 400 antes each.

Without any action abstraction, using antes of 1 for each player this game has approximately 10^{11} states and 10^{11} action sequences. Due to the large number of action sequences, state of the art ϵ-Nash equilibrium solvers are unable to solve the game directly, as it would take up at least 1480 gigabytes of memory. The abstractions we are looking for allow for one bet after every different sequence of actions, producing games with 12 bets for each player. These abstractions have only 1800 states and 4518 action sequences, with the multiplayer betting version of them having just 24 more states - one for each bet. This illustrates one of the strengths of our approach; by considering bet sizes over a range, our algorithm uses a small number of states to search over a large portion of the full game's state space, constantly moving the bet sizes in a direction that increases overall utility for the player making the bet.

A typical run of the CFR algorithm consists of 200 million iterations or more to ensure convergence. For the abstract games we created, it took only 15 million iterations for the best response values to stabilize. When run for more iterations the game value of the abstractions improved very little. Our algorithm runs slower on the multiplayer betting games than CFR does on an equivalent sized abstraction, as there are more action sequences. We run the algorithm until we see diminishing returns, in this case 15 million iterations, and then create the appropriate abstraction and run CFR on that game up to the desired level of convergence. In all three cases we ran CFR for one billion iterations on the abstractions to obtain very tight bounds. We obtained game values of -0.060 antes for player one's new betting abstraction, -0.072 for the baseline abstraction, and -0.078 for player two's new betting abstraction. The abstractions chosen for player one and two represent respective 12% and 7.7% improvements over the baseline abstraction. Game values are expressed as utilities for player one, so a smaller value is better for player two.

Consider the bet sizes used in these new abstractions. If a bet is never made, or the opponent always folds when the bet is made, then the size of the bet does not affect the utility. This was the case for two of the bets for player one, and four of the bets for player two. Table 1 shows the ranges in which the other 18 bets lie. We divided the $L = 0.8$ and $H = 1.5$ range into seven ranges of width 0.1, and as we can see there are a large number of different bet sizes used, one or more in every one of these 0.1 width ranges, with the exception of $1.3 - 1.4$. This illustrates the complexity of the abstractions we found - a wide range of bet sizes are used, something which is difficult to model using expert knowledge.

Figure 3 shows a graph of three bet sizes chosen by the algorithm over time. Each bet size is relatively stable after 15 million iterations, which is consistent with the fact that the game values of the new abstractions do not improve much beyond this point. The action sequences leading up to these bets are given in the legend of Figure 3 and are represented as follows: "c" stands for check/call, "b" is bet and "r" is raise. The "/" indicates the dealing of the community card. The fact that the bet size made after the "cbrc/c" sequence is smaller than the other two agrees with our expectations based on poker theory. This is because of the three

Range	# Bets
$0.8 - 0.9$	3
$0.9 - 1.0$	1
$1.0 - 1.1$	3
$1.1 - 1.2$	4
$1.2 - 1.3$	1
$1.3 - 1.4$	0
$1.4 - 1.5$	6

Table 1: Distribution of bet sizes found.

sequences this one represents the most threat from the opponent: they check-raised before the community card, which represents a stronger hand than simply betting, as they do in the other two sequences. In general the stronger the opponent's distribution of possible hands is relative to ours, the smaller we expect our bet size should be.

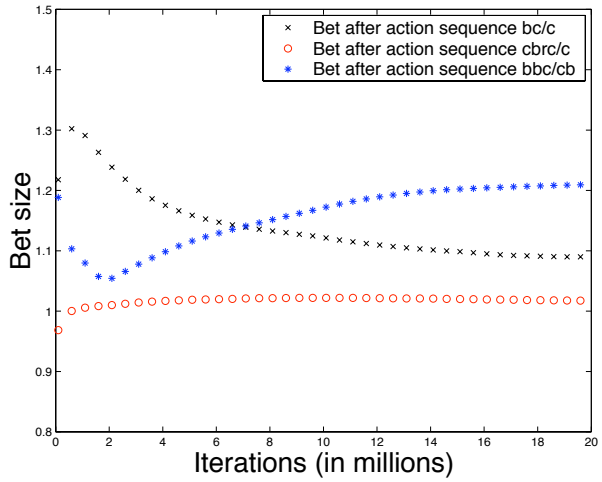


Figure 3: Three different bet sizes as determined by our algorithm over a run of 20 million iterations.

Finding multiplayer betting games strategies

Consider repeated playing of an extensive-form game. Let $\sigma^t(I,a)$ be the probability of taking action a at information set I on iteration t. $A(I)$ is the set of actions available at information set I. $R_i^t(I,a)$ is the regret for player i of not taking action a at information set I, and $R_i^+(I,a) = \max(R_i^t(I,a), 0)$. Regret matching chooses actions at each information set as follows (Zinkevich et al. 2008):

$$\sigma_i^{t+1}(I,a) = \begin{cases} \frac{R_i^+(I,a)}{\sum_{a \in A(I)} R_i^+(I,a)} & \text{if } \sum_{a \in A(I)} R_i^+(I,a) > 0 \\ \frac{1}{|A(I)|} & \text{otherwise.} \end{cases} \quad (9)$$

CFR uses regret matching to choose actions for every player at every information set. Due to Theorem 1 we know that if we can minimize regret for players in the abstract game given by setting $s_i = B(\overline{P(H)}_i^T)$, where $\overline{P(H)}_i^T$ is the average of all $P(H)_i^t$ choices made by bet sizing player i up to time $t = T$, then regret will be minimized for players one and two in the multiplayer betting game as well. We tried applying CFR directly to the multiplayer betting version of some small poker games, but it did not converge. When CFR is applied to a two player zero-sum game the average of the strategy profiles used each iteration up to time T converges, but the individual strategy profiles used on each iteration do not. With this in mind we decided to try continuing to use Equation 9 to minimize regret for players one and two, but using a smoother regret minimizing algorithm to choose new values of $P(H)_i^t$ each iteration, moving each effective bet size $s_i^t = B(P(H)_i^t)$ small amounts in a direction that minimizes regret for player i. There are many different possible ways to achieve this, and of all the ones we tested the following performed the best:

$$s_i^{t+1} = \begin{cases} \frac{(t-1)s_i^{t-1} + sr_i^t}{t} & \text{if } t < N \\ \frac{(N-1)s_i^{t-1} + sr_i^t}{N} & \text{otherwise} \end{cases} \quad (10)$$

where s_i^t is the effective bet size player i plays on iteration t, sr_i^t is the effective bet size player i would play on iteration t if we used Equation 9 to choose $P(H)_i^t$, and N is a constant. Having s_i^{t+1} we combine Equation 1 with Equation 10 to determine $P(H)_i^{t+1}$. We can see that up to iteration $t = N$ the effective bet size chosen by this algorithm is simply the average of all the effective bet sizes that Equation 9 would have picked each iteration. Using this average makes the movement of the effective bet size from iteration to iteration much smoother. After time $t = N$ we still use the average of all past choices of regret matching, but we weight the effective bet size of the latest choice by $1/N$. This ensures that the effective bet size continues to move some minimal amount in a direction that minimizes regret.

This algorithm produced an ϵ-Nash equilibrium for multiplayer betting half-street Kuhn poker using $N = \infty$. In larger Kuhn games, however, $N = \infty$ causes the effective bet size played each iteration to converge before the average strategies for players one and two. We found that a setting of $N = 10000$ for all bet sizing players worked well in all test cases, and this is the value used to produce all of the results presented earlier. Tuning of this parameter should lead to improved performance, and we will consider this as we apply our technique to more and more complex domains.

Conclusion

The problem of effectively abstracting the action space of large extensive form games has been largely ignored. Agents created to operate in huge domains such as no-limit poker, where the action space is much larger than the state space, have typically tried to avoid this problem by using simple hand crafted abstractions. Researchers have instead focused on developing ϵ-Nash solvers as well as techniques to abstract the state space of the game. Our results complement that research by presenting a technique that can be applied to find quality abstractions of the action space.

We have shown that applying a transformation to small poker games and running a new regret minimizing algorithm generates abstractions that have the same game value as the game they represent. We showed that applying this to larger poker games for which we have no analytical solutions, we can obtain utility gains for both players over the game value of a standard baseline abstraction. These improvements in game value, coupled with the range of different bet sizes used in the new abstractions, demonstrate that our technique is capable of generating complex action abstractions, and that such complexity leads to a notable increase in utility over simple hand-crafted abstractions. Our next step is to apply this technique to full-scale no-limit Texas Hold'em.

Acknowledgements

We would like to thank all the members of the University of Alberta Computer Poker Research Group for their valuable insights and support. This research was supported in part by research grants from the Natural Sciences and Engineering Research Council of Canada (NSERC), the Alberta Informatics Circle of Research Excellence (iCORE) and Alberta Ingenuity through the Alberta Ingenuity Centre for Machine Learning.

References

Antos, A.; Munos, R.; and Szepesvari, C. 2007. Fitted Q-iteration in continuous action-space MDPs. In *NIPS*, 9–16.

Chen, B., and Ankenman, J. 2007. *The mathematics of poker*. ConJelCo publishing company.

Dean, T.; Kim, K.; and Givan, R. 1998. Solving stochastic planning problems with large state and action spaces. In *Proc. Fourth International Conf. on Artificial Intelligence Planning Systems*, 102–110. AAAI Press.

Gilpin, A., and Sandholm, T. 2007a. Better automated abstraction techniques for imperfect information games, with application to Texas Hold'em poker. In *AAMAS*, 1168–1175.

Gilpin, A., and Sandholm, T. 2007b. Gradient-based algorithms for finding Nash equilibria in extensive form games. In *WINE*, 57–69.

Gilpin, A.; Sandholm, T.; and Sorensen, T. B. 2007. Potential-aware automated abstraction of sequential games, and holistic equilibrium analysis of Texas Hold'em poker. In *AAAI*, 50–57.

Gilpin, A.; Sandholm, T.; and Sorensen, T. B. 2008. A heads-up no-limit Texas Hold'em poker player: Discretized betting models and automatically generated equilibrium-finding programs. In *AAMAS*, 911–918.

Kuhn, H. W. 1950. Simplified two-person poker. *Contributions to the Theory of Games* 1:97–103.

Lazaric, A.; Restelli, M.; and Bonarini, A. 2008. Reinforcement learning in continuous action spaces through sequential Monte Carlo methods. In *NIPS*, 833–840.

Madeira, C. A. G.; Corruble, V.; and Ramalho, G. 2006. Designing a reinforcement learning-based adaptive AI for large-scale strategy games. In *AIIDE*, 121–123.

Schnizlein, D.; Bowling, M.; and Szafron, D. 2009. Probabilistic state translation in extensive games with large action sets. In *IJCAI*.

Schnizlein, D. 2009. State translation in no-limit poker. Master's thesis, University of Alberta.

Tijs, S. 1980. Stochastic games with one big action space in each state. *Methods of Operations Research* 38:161–173.

van Hasselt, H., and Wiering, M. 2007. Reinforcement learning in continuous action spaces. In *IEEE Symposium on Approximate Dynamic Programming and Reinforcement Learning*, 272–279.

Waugh, K.; Bard, N.; and Bowling, M. 2009. Strategy grafting in extensive games. In *NIPS*, 2026–2034.

Waugh, K.; Schnizlein, D.; Bowling, M.; and Szafron, D. 2009a. Abstraction pathologies in extensive games. In *AAMAS*, 781–788.

Waugh, K.; Zinkevich, M.; Johanson, M.; Kan, M.; Schnizlein, D.; and Bowling, M. 2009b. A practical use of imperfect recall. In *Proceedings of the Eighth Symposium on Abstraction, Reformulation and Approximation (SARA)*, 175–182.

Zinkevich, M.; Johanson, M.; Bowling, M.; and Piccione, C. 2008. Regret minimization in games with incomplete information. In *NIPS*, 1729–1736.