Self-Play Monte-Carlo Tree Search in Computer Poker

Johannes Heinrich and **David Silver**
University College London
Gower Street, London, UK
{j.heinrich,d.silver}@cs.ucl.ac.uk

Abstract

Self-play reinforcement learning has proved to be successful in many perfect information two-player games. However, research carrying over its theoretical guarantees and practical success to games of imperfect information has been lacking. In this paper, we evaluate self-play Monte-Carlo Tree Search (MCTS) in limit Texas Hold'em and Kuhn poker. We introduce a variant of the established UCB algorithm and provide first empirical results demonstrating its ability to find approximate Nash equilibria.

Introduction

Reinforcement learning has traditionally focused on stationary single-agent environments. Its applicability to fully observable multi-agent Markov games has been explored by (Littman 1996). Backgammon and computer Go are two examples of fully observable two-player games where reinforcement learning methods have achieved outstanding performance (Tesauro 1992; Gelly et al. 2012). Computer poker provides a diversity of stochastic imperfect information games of different sizes and has proved to be a fruitful research domain for game theory and artificial intelligence (Sandholm 2010; Rubin and Watson 2011). Therefore, it is an ideal research subject for self-play reinforcement learning in partially observable multi-agent domains.

Game-theoretic approaches have played a dominant role in furthering algorithmic performance in computer poker. The game is usually formulated as an extensive-form game with imperfect information and most attention has been directed towards the two-player Texas Hold'em variant. Abstracting the game (Billings et al. 2003; Gilpin and Sandholm 2006), by combining similar nodes, reduces the size of the game tree and has allowed various approaches to compute approximate Nash equilibria. Common game-theoretic methods in computer poker are based on linear programming (Billings et al. 2003; Gilpin and Sandholm 2006), non-smooth convex optimization (Hoda et al. 2010), fictitious play (Ganzfried and Sandholm 2008) or counterfactual regret minimization (CFR) (Zinkevich et al. 2007). Except for Monte-Carlo counterfactual regret minimization (MCCFR) (Lanctot et al. 2009), these are generally full-width methods and therefore are particularly exposed to the *curse of dimensionality*. This might prevent them from scaling to larger games, e.g. multi-player poker. In addition, some methods are undefined for games with more than two players, e.g. the excessive gap technique (Hoda et al. 2010), while other approaches lose their theoretical guarantees, e.g. CFR. In spite of this, CFR has achieved strong performance in three-player limit Texas Hold'em (Risk and Szafron 2010). The MCTS methods discussed in this paper are well defined for any number of players but do not yet have any established theoretical convergence guarantees even for two-player zero-sum imperfect information games.

MCTS is a simulation-based search algorithm that has been successful in high-dimensional domains, e.g. computer Go (Gelly et al. 2012). Selective sampling of trajectories of the game enables it to prioritise the most promising regions of the search space. Furthermore, it only requires a black box simulator and might therefore plan in complex environments, e.g. with approximate generative models of real world applications. Finally, it is built on principles of reinforcement learning and can therefore leverage its well-developed machinery. For example, function approximation and bootstrapping are two ideas that can dramatically improve the efficiency of learning algorithms (Tesauro 1992; Silver, Sutton, and Müller 2012).

(Ponsen, de Jong, and Lanctot 2011) compared the quality of policies found by MCTS and MCCFR in computer poker. They concluded that MCTS quickly finds a good but suboptimal policy, while MCCFR initially learns more slowly but converges to the optimal policy over time.

In this paper, we extend the evaluation of MCTS-based methods in computer poker. We introduce a variant of UCT, Smooth UCT, that combines MCTS with elements of fictitious play and provide first empirical results of its convergence to Nash equilibria in Kuhn poker. This algorithm might address the inability of UCT to converge close to a Nash equilibrium over time, while retaining UCT's fast initial learning rate. Both UCT and Smooth UCT achieve high performance against competitive approximate Nash equilibria in limit Texas Hold'em. This demonstrates that strong policies can be learned from UCT-based self-play methods in partially observable environments.

Background

Extensive-Form Games

Extensive-form games are a rich model of multi-agent interaction. The representation is based on a game tree. Introducing partitions of the tree that constrain the players' information about the exact position in the game tree allows to add versatile structure to the game, e.g. simultaneous moves and partial observability. Formally, based on the definition of (Selten 1975),

Definition 1. An extensive-form game with imperfect information consists of

- A set of players $N = \{1, ..., n\}$ and a chance player c.
- A set of states $\mathcal{S}$ corresponding to nodes in a rooted game tree with initial state s_0. $\mathcal{S}_Z \subset \mathcal{S}$ is the set of terminal states.
- A partition of the state space into player sets P^i, $i \in N \cup \{c\}$. Each state $s \in P^i$ represents a decision node of player i and $A(s)$ is the set of actions available to this player in s. In particular, P^c is the set of states at which chance determines the successor state.
- For each player a set of information states $\mathcal{O}^i$ and a surjective information function $\mathcal{I}^i$ that assigns each state $s \in P^i$ to an information state $o \in \mathcal{O}^i$. For any information state $o \in \mathcal{O}^i$ an information set $I^i = (\mathcal{I}^i)^{-1}(o)$ is a set of states $s \in P^i$ that are indistinguishable for player i.
- A chance function $\mathcal{T}^c(s, a) = \mathbb{P}(a_t^c = a \mid s_t = s)$, $s \in P^c$, that determines chance events at chance nodes. Transitions at players' decision nodes are described by their policies.
- For each player a reward function R^i that maps terminal states to payoffs.

For each player $i \in N$ the sequence of his information states and own actions in an episode forms a history $h_t^i = \{o_1^i, a_1^i, o_2^i, a_2^i, ..., o_t^i\}$. A game has perfect recall if each player's current information state o_t^i implies knowledge of his whole history h_t^i of the episode. If a player only knows some strict subset of his history h_t^i, then the game has imperfect recall.

The following definition is similar to (Waugh et al. 2009) but makes use of information functions.

Definition 2. Let Γ be an extensive-form game. An abstraction of Γ is a collection of surjective functions $f_A^i : \mathcal{O}^i \to \tilde{\mathcal{O}}^i$ that map the information states of Γ to some alternative information state spaces $\tilde{\mathcal{O}}^i$. By composition of the abstraction with the information functions $\mathcal{I}^i$ of Γ, we obtain alternative information functions, $\tilde{\mathcal{I}}^i = f_A^i \circ \mathcal{I}^i$, that induce an abstracted extensive-form game.

Mapping multiple information states to a single alternative information state results in the affected player not being able to distinguish between the mapped states.

Each player's behavioural strategy is determined by his policy $\pi^i(o, a) = \mathbb{P}(a_t^i = a \mid o_t^i = o)$, which is a probability distribution over actions given an information state, and Δ^i is the set of all policies of player i. A policy profile $\pi = (\pi^1, ..., \pi^n)$ is a collection of policies for all players. π^{-i} refers to all policies in π except π^i. $R^i(\pi)$ is the expected reward of player i if all players follow the policy profile π. The set of best responses of player i to his opponents' policies π^{-i} is $b^i(\pi^{-i}) = \arg\max_{\pi^i \in \Delta^i} R^i(\pi^i, \pi^{-i})$. For $\epsilon > 0$, $b_\epsilon^i(\pi^{-i}) = \{\pi^i \in \Delta^i : R^i(\pi^i, \pi^{-i}) \geq R^i(b^i(\pi^{-i}), \pi^{-i}) - \epsilon\}$ defines the set of ϵ-best responses to the policy profile π^{-i}.

Definition 3. A Nash equilibrium of an extensive-form game is a policy profile π such that $\pi^i \in b^i(\pi^{-i})$ for all $i \in N$. An ϵ-Nash equilibrium is a policy profile π such that $\pi^i \in b_\epsilon^i(\pi^{-i})$ for all $i \in N$.

MCTS

MCTS (Coulom 2006) is a simulation-based search algorithm. It is able to plan in high-dimensional environments by sampling episodes through Monte-Carlo simulation. These simulations are guided by an action selection mechanism that explores the most promising regions of the state space. This guided search results in efficient, asymmetric search trees. MCTS converges to optimal policies in fully observable Markovian environments.

A MCTS algorithm requires the following components. Given a state and action, a black box simulator of the game samples a successor state and reward. A learning algorithm uses simulated trajectories and outcomes to update some statistics of the visited nodes in the search tree. A tree policy is defined by an action selection mechanism that chooses actions based on a node's statistics. A rollout policy determines default behaviour for states that are out of the scope of the search tree.

For a specified amount of planning time the algorithm repeats the following. It starts each Monte-Carlo simulation at the root node and follows its tree policy until either reaching a terminal state or the boundary of the search tree. Leaving the scope of the search tree, the rollout policy is used to play out the simulation until reaching a terminal state. In this case, we expand our tree by a state node where we have left the tree. This approach selectively grows the tree in areas that are frequently encountered in simulations. After reaching a terminal state, the rewards are propagated back so that each visited state node can update its statistics.

Common MCTS keeps track of the following node values. $N(s)$ is the number of visits by a Monte-Carlo simulation to node s. $N(s, a)$ counts the number of times action a has been chosen at node s. $Q(s, a)$ is the estimated value of choosing action a at node s. The action value estimates are usually updated by Monte-Carlo evaluation, $Q(s, a) = \frac{1}{N(s,a)} \sum_{i=0}^{N(s,a)} R_s(i)$, where $R_s(i)$ is the cumulative discounted reward achieved after visiting state s in the i-th Monte-Carlo simulation that has encountered s.

(Kocsis and Szepesvári 2006) suggested using the bandit-based algorithm UCB (Auer, Cesa-Bianchi, and Fischer 2002) to select actions in the Monte-Carlo search tree. The resulting MCTS method, Upper Confidence Trees (UCT), selects greedily between action values that have been enhanced by an exploration bonus,

$$\pi_{tree}(s) = \arg\max_a Q(s,a) + c\sqrt{\frac{\log N(s)}{N(s,a)}}. \quad (1)$$

The exploration bonus parameter c adjusts the balance between exploration and exploitation. For suitable c the probability of choosing a suboptimal action has been shown to converge to 0 (Kocsis and Szepesvári 2006). The appropriate scale of c depends on the size of the rewards.

Other MCTS methods have been proposed in the literature. (Silver and Veness 2010) adapt MCTS to partially observable Markov decision processes (POMDPs) and prove convergence given a true initial belief state. (Auger 2011) and (Cowling, Powley, and Whitehouse 2012) study MCTS in imperfect information games and use the bandit method EXP3 (Auer et al. 1995) for action selection.

Self-Play MCTS in Extensive-Form Games

Our goal is to use MCTS-based planning to learn an optimal policy profile of an extensive-form game with imperfect information.

Planning in a multi-player game requires simulation of players' behaviour. One approach might be to assume explicit models of player behaviour and then use a planning algorithm to learn a best response against these models. However, the quality of the policy profile learned with this approach will depend on the assumed player models. In particular, to learn an optimal policy profile, we will most likely need a player model that produces close to optimal behaviour in the first place. We will therefore pursue an alternative approach that does not require this type of prior knowledge. Self-play planning does not assume any explicit models of player behaviour. Instead, it samples players' actions from the current policy profile, that is suggested by the planning process. E.g. self-play MCTS samples player behaviour from the players' tree policies.

The asymmetry of information in an extensive-form game with imperfect information generally does not allow us to span a single collective search tree. We will therefore describe a general algorithm that uses a separate search tree for each player. For each player we grow a tree T^i over his information states $\mathcal{O}^i$. $T^i(o^i)$ is the node in player i's tree that represents his information state o^i. In a game with perfect recall a player remembers his sequence of previous information states and actions $h^i_t = \{o^i_1, a^i_1, o^i_2, a^i_2, ..., o^i_t\}$. By definition of an extensive-form game, this knowledge is also represented in the information state o^i_t. Therefore, T^i is a proper tree. An imperfect recall extensive-form game yields a recombining tree.

Algorithm 1 describes MCTS that has been adapted to the multi-player, imperfect information setting of extensive-form games. The game mechanics are sampled from transition and reward simulators $\mathcal{G}$ and $\mathcal{R}$. The transition simulator takes a state and action as inputs and generates a sample of a successor state $s_{t+1} \sim \mathcal{G}(s_t, a^i_t)$, where the action a^i_t belongs to the player i who makes decisions at state s_t. The reward simulator generates all players' payoffs at terminal states, i.e. $\boldsymbol{r_T} \sim \mathcal{R}(s_T)$. The information function, $\mathcal{I}^i(s)$, determines the acting player i's information state. OUT-OF-TREE keeps track of which player has left the scope of his search tree in the current episode.

The basic idea of searching trees over information states is similar to (Auger 2011; Cowling, Powley, and Whitehouse 2012). However, algorithm 1 defines a more general class of MCTS-based algorithms that can be specified by their action selection and node updating functions. These functions are responsible to sample from and update the tree policy. In this work, we focus on UCT-based methods.

Algorithm 1 Self-play MCTS in extensive-form games

```
function SEARCH($\Gamma$)
    while within computational budget do
        $s_0 \sim \Gamma$
        SIMULATE($s_0$)
    end while
    return $\pi_{tree}$
end function

function ROLLOUT($s$)
    $a \sim \pi_{rollout}(s)$
    $s' \sim \mathcal{G}(s,a)$
    return SIMULATE($s'$)
end function

function SIMULATE($s$)
    if ISTERMINAL($s$) then
        return $\boldsymbol{r} \sim \mathcal{R}(s)$
    end if
    $i$ = PLAYER($s$)
    if OUT-OF-TREE($i$) then
        return ROLLOUT($s$)
    end if
    $o^i = \mathcal{I}^i(s)$
    if $o^i \notin T^i$ then
        EXPANDTREE($T^i, o^i$)
        $a \sim \pi_{rollout}(s)$
        OUT-OF-TREE(i) $\leftarrow$ true
    else
        $a$ = SELECT($T^i(o^i)$)
    end if
    $s' \sim \mathcal{G}(s,a)$
    $\boldsymbol{r} \leftarrow$ SIMULATE($s'$)
    UPDATE($T^i(o^i), a, r^i$)
    return $\boldsymbol{r}$
end function
```

Information State UCT (IS-UCT)

IS-UCT uses UCB to select from and update the tree policy in algorithm 1. It can be seen as a multi-player version of Partially Observable UCT (PO-UCT) (Silver and Veness 2010), that searches trees over histories of information states instead of histories of observations and actions of a POMDP. In fact, assuming fixed opposing players' policies, a player is effectively acting in a POMDP. However, in self-play planning, player's policies might never become stationary and keep changing indefinitely. Indeed, (Ponsen, de Jong, and Lanctot 2011) empirically show the inability of UCT to converge to a Nash equilibrium in Kuhn poker. However, it seems to unlearn dominated actions and therefore might yield decent but potentially exploitable performance in practice.

Smooth IS-UCT

To address the lack of convergence guarantees of IS-UCT in partially observable self-play environments, we introduce an alternative tree policy algorithm. It has similarities to generalised weakened fictitious play (Leslie and Collins 2006), for which convergence results have been established for some types of games, e.g. two-player zero-sum games.

Definition 4. A Smooth UCB process is defined by the following sequence of policies:

$$\begin{aligned}\pi^i_t &= (1-\eta_t)\bar{\pi}^i_{t-1} + \eta_t \mathrm{UCB}^i_t,\\ a^i_t &\sim \pi^i_t\\ \bar{\pi}^i_t &= \bar{\pi}^i_{t-1} + \frac{1}{t}(a^i_t - \bar{\pi}^i_{t-1}) \end{aligned} \qquad (2)$$

for some time t adapted sequence $\eta_t \in [c, 1]$, $c > 0$. π_t is the behavioural policy profile that is used for planning at simulation t. Players' actions, a^i_t, $i \in N$, are sampled from this policy. $\bar{\pi}_t$ is the average policy profile at time t.

UCB plays a greedy best response towards its optimistic action value estimates. Smooth UCB plays a smooth best response instead. If UCB were an ϵ-best response to the opponents' average policy profile, $\bar{\pi}^{-i}_t$, with $\epsilon \to 0$ as $t \to \infty$, then the Smooth UCB process would be a generalised weakened fictitious play. E.g. this would be the case if the action value estimates were generated from unbiased samples of the average policy. However, this is clearly not what happens in self-play, where at each time players are trying to exploit each other rather than playing their average policy. Intuitively, mixing the current UCB policy with the average policy might help with this policy evaluation problem.

Algorithm 2 instantiates general self-play MCTS with a Smooth UCB tree policy. For a constant $\eta_t = 1$, we obtain IS-UCT as a special case. We propose two schedules to set η_t in the experiments section. Note that for a cheaply determined η_t the update and selection steps of Smooth UCB do not have any overhead compared to UCB.

We are currently working on developing the theory of this algorithm. Our experiments on Kuhn poker provide first empirical evidence of the algorithm's ability to find approximate Nash equilibria.

Algorithm 2 Smooth IS-UCT

```
SEARCH($\Gamma$), SIMULATE($s$) and ROLLOUT($s$) as in algorithm 1

function SELECT($T^i(o^i)$)
    $u \sim U[0,1]$
    if $u < \eta_t$ then
        return $\arg\max_a Q(o^i,a) + c\sqrt{\frac{\log N(o^i)}{N(o^i,a)}}$
    else
        $\forall a \in A(o^i) : p(a) \leftarrow \frac{N(o^i,a)}{N(o^i)}$
        return $a \sim p$
    end if
end function

function UPDATE($T^i(o^i), a, r^i$)
    $N(o^i) \leftarrow N(o^i) + 1$
    $N(o^i,a) \leftarrow N(o^i,a) + 1$
    $Q(o^i,a) \leftarrow Q(o^i,a) + \frac{r^i - Q(o^i,a)}{N(o^i,a)}$
end function
```

Experiments

We evaluate IS-UCT and Smooth IS-UCT in Kuhn and Texas Hold'em poker games.

Kuhn Poker

Kuhn poker (Kuhn 1950) is a small extensive-form game with imperfect information. It features typical elements of poker, e.g. balancing betting frequencies for different types of holdings. However, it does not include community cards and multi-street play. As there exist closed-form game-theoretic solutions, it is well suited for a first overview of the performance and calibration of different methods.

(Ponsen, de Jong, and Lanctot 2011) tested UCT against MCCFR in Kuhn poker. We conducted a similar experiment with Smooth IS-UCT. We calibrated the exploration parameter for both IS-UCT and Smooth IS-UCT, and set it to 2 and 1.75 respectively. We tested two choices of Smooth UCB's mixing parameter sequence η_t:

$$\eta^i_t(o^i) = \max_a \bar{\pi}^i_{t-1}(o^i,a) \qquad (3)$$

$$\eta^i_t(o^i) = 0.9\frac{10000}{10000 + \sqrt{N_{t-1}(o^i)}} \qquad (4)$$

Both worked well in standard Kuhn poker. However, only schedule (3) yielded stable performance across several variants of Kuhn poker with varying initial pot size. All reported results for Smooth IS-UCT on Kuhn poker were generated with schedule (3). The average policies' squared (SQ-ERR) and dominated (DOM-ERR) errors were measured after every 10^4 episodes. The squared error of a policy is its mean-squared error measured against the closest Nash equilibrium. The dominated error of a policy is the sum of the probabilities with which a dominated action is taken at an information state. The results, shown in figures 1 and 2, were averaged over 10 identically repeated runs of the experiment.

In standard Kuhn poker, Smooth IS-UCT performs better both in terms of squared and dominated error. We repeated the experiment for variants of Kuhn poker with higher initial pot sizes. This is supposed to emulate the varying optimal action probabilities that in poker depend on the bet size relative to the pot. Using mixing schedule (3), Smooth IS-UCT was able to converge close to a Nash equilibrium in each variant. For larger pot sizes, Smooth IS-UCT seems to suffer a larger dominated error than IS-UCT.

To conclude, Smooth IS-UCT might focus its self-play search on regions of the policy space that are close to a Nash

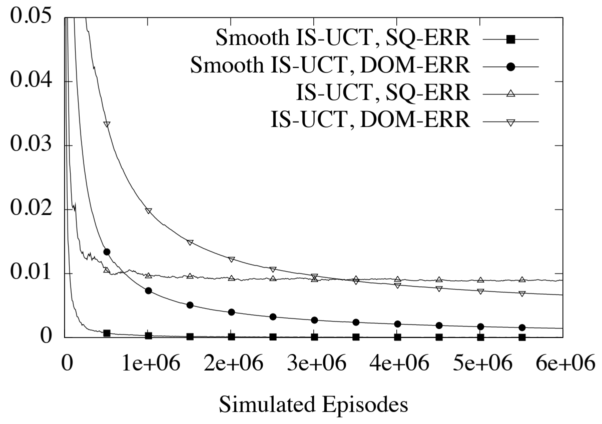

Figure 1: Learning curve of IS-UCT and Smooth IS-UCT in Kuhn poker. The x-axis denotes the number of simulated episodes. The y-axis denotes the quality of a policy in terms of its dominated and squared error measured against a closest Nash equilibrium.

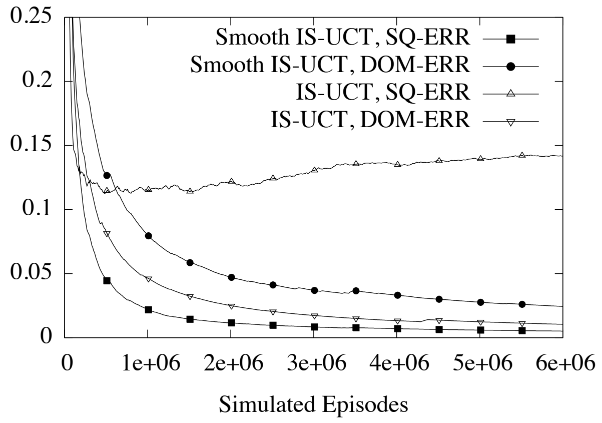

Figure 2: Learning curve of IS-UCT and Smooth IS-UCT in a Kuhn poker variant with an initial pot size of 8.

equilibrium. However, in large games, the number of visits to a node might be too low to provide a stochastic average policy with small dominated error.

Limit Texas Hold'em

We consider limit Texas Hold'em with two and three players. The two-player variants' game tree contains about 10^{18} nodes. In order to reduce the game tree to a tractable size, various abstraction techniques have been proposed in the literature (Billings et al. 2003; Gilpin and Sandholm 2006; Johanson et al. 2013). The most common approach is to group information states according to strategic similarity of the corresponding cards and leave the players' action sequences unabstracted.

A player's information state can be described as a tuple $o_t^i = (\bar{\boldsymbol{a}}_t, \boldsymbol{c}_t, \boldsymbol{p}_t^i)$, where $\bar{\boldsymbol{a}}_t$ is the sequence of all players' actions except chance, $\boldsymbol{p}_t^i$ are the two private cards held by player i and $\boldsymbol{c}_t$ is the sequence of community cards publicly revealed by chance by time t. In Texas Hold'em only the private cards constitute hidden information, i.e. each player observes everything except his opponents' private cards.

Let $g_A(\boldsymbol{c}, \boldsymbol{p}^i) \in \mathbb{N}$ be a function that maps tuples of public community cards and private cards to an abstraction bucket. The function $f_A(\bar{\boldsymbol{a}}, \boldsymbol{c}, \boldsymbol{p}^i) = (\bar{\boldsymbol{a}}, g_A(\boldsymbol{c}, \boldsymbol{p}^i))$ defines an abstraction that maps information states $\mathcal{O} = \cup_{i \in N} \mathcal{O}^i$ to alternative information states $\tilde{\mathcal{O}} = \cup_{i \in N} \tilde{\mathcal{O}}^i$ of an abstracted game. An abstracted information state is a tuple $(\bar{\boldsymbol{a}}, n)$ where $\bar{\boldsymbol{a}}$ is the unabstracted sequence of all players' actions except chance and $n \in \mathbb{N}$ identifies the abstraction bucket.

In this work, we use a metric called expected hand strength squared (Zinkevich et al. 2007). At a final betting round and given the corresponding set of community cards, the hand strength of a holding is defined as its winning percentage against all possible other holdings. On any betting round, $\mathbb{E}[\text{HS}^2]$ denotes the expected value of the squared hand strength on the final betting round. We have discretised the resulting $\mathbb{E}[\text{HS}^2]$ values with an equidistant grid over their range, $[0, 1]$. Table 1 shows the grid sizes that we used in our experiments. We used an imperfect recall abstraction that does not let a player remember his $\mathbb{E}[\text{HS}^2]$ values of previous betting rounds, i.e. $g_A(\boldsymbol{c}, \boldsymbol{p}^i)$ is set to the thresholded $\mathbb{E}[\text{HS}^2]$ value of $\boldsymbol{p}^i$ given $\boldsymbol{c}$.

Game size	Preflop	Flop	Turn	River
2p S	169	100	100	100
2p L	169	1000	500	200
3p	169	1000	100	10

Table 1: Abstraction bucket discretisation grids used in our two and three-player Texas Hold'em experiments

In limit Texas Hold'em, there is an upper bound on the possible terminal pot size based on the betting that has occurred so far in the episode. This is because betting is capped at each betting round. To avoid unnecessary exploration we update the exploration parameter for the current episode at the beginning of each betting round and set it to

$$c_t = \text{potsize} + k * \text{remaining betting potential}, \qquad (5)$$

where $k \in [0, 1]$ is a constant parameter and the remaining betting potential is the maximum possible amount that players can add to the pot in the remainder of the episode.

Each two-player policy was trained for about 7 billion episodes, which took about 30 hours on a modern computer without using parallelization. Each three-player agent was trained for about 12 billion episodes, requiring about 48 hours of training time. After training, the policies were frozen to return the greedy, i.e. highest value, action at each information state. Performance is measured in milli-big-blinds per hand, mb/h.

Two-player We evaluated the performance of the trained policies against benchmark opponents in two-player limit Texas Hold'em. Both opponents, Sparbot (Billings et al.

2003) and FellOmen2, play approximate Nash equilibrium strategies. Sparbot is available in the software Poker Academy Pro. FellOmen2 is a publicly available poker bot that was developed by Ian Fellows. It plays an approximate Nash equilibrium that was learned from fictitious play. It placed second, tying three-way, in the AAAI-08 Computer Poker Competition.

All two-player MCTS policies were trained using exploration schedule (5), with parameter k given in the table. The Smooth IS-UCT agents used the same mixing schedule (3) as in Kuhn poker. The abstraction sizes are shown in table 1.

The results in table 2 show an overall positive performance of UCT-based methods. However, some policies' performance is not stable against both opponents. This suggests that these policies might not be a good approximation of a Nash equilibrium. Note that we froze the policies after training and that these frozen policies are deterministic. This generally yields exploitable policies, which in turn exploit some types of policies. This might explain some of the contrary results, i.e. very positive performance against Sparbot while losing a large amount against FellOmen2. A coarse abstraction might emphasize these effects of freezing policies, as larger amounts of information states would be set to a particular action.

Due to the large standard errors relative to the differences in performance, comparing individual results is difficult. IS-UCT with a coarse abstraction and low exploration parameter, IS-UCT S, k=0.5, trained the policy that achieves the most robust performance against both opponents. On the other hand, Smooth IS-UCT trained two policies, with both fine and coarse abstractions, that achieve robust performance. More significant test results and an evaluation of performance for different training times are required to better compare Smooth IS-UCT against IS-UCT.

	Sparbot	FellOmen2
Coarse abstraction:		
IS-UCT S, k=1	72 ± 21	-47 ± 20
IS-UCT S, k=0.5	54 ± 21	26 ± 21
Smooth IS-UCT S, k=1	51 ± 20	7 ± 21
Smooth IS-UCT S, k=0.5	56 ± 20	-41 ± 20
Fine abstraction:		
IS-UCT L, k=0.5	60 ± 20	-17 ± 21
Smooth IS-UCT L, k=0.5	18 ± 19	32 ± 21

Table 2: Two-player limit Texas Hold'em winnings in mb/h and their 68% confidence intervals. Each match-up was run for at least 60000 hands.

Three-player In three-player limit Texas Hold'em, we tested the trained policies against benchmark and dummy opponents. Poki (Billings 2006) is a bot available in Poker Academy Pro. Its architecture is based on a mixture of technologies, including expert knowledge and online Monte-Carlo simulations. It won the six-player limit Texas Hold'em event of the AAAI-08 Computer Poker Competition. Due to a lack of publicly available benchmark bots capable of playing three-player limit Texas Hold'em, we also included match-ups against dummy agents that blindly play according to a fixed probability distribution. We chose two dummy agents that were suggested by (Risk and Szafron 2010). The Always-Raise agent, AR, always plays the bet/raise action. The Probe agent randomly chooses between the bet/raise and check/call action with equal probability.

All three-player MCTS policies were trained with exploration schedule (5) with $k = 0.5$. Smooth IS-UCT used mixing schedule (3), with η_t set to zero for nodes that have been visited less than 10000 times in order to avoid disproportionate exploration at rarely visited nodes.

The results, shown in table 3, suggest that self-play MCTS can train strong three-player policies. IS-UCT outperforms Smooth IS-UCT in the match-up against two instances of Poki. It performs slightly worse than Smooth IS-UCT in the other two match-ups. Both MCTS policies achieve strong results against the combination of AR and Probe despite using imperfect recall. This is notable because (Risk and Szafron 2010) reported anomalous, strongly negative results against this combination of dummy agents when using imperfect recall with CFR.

Match-up	IS-UCT	Smooth IS-UCT
2x Poki	163 ± 19	118 ± 19
Poki, IS-UCT	-	81 ± 24
Poki, Smooth IS-UCT	73 ± 24	-
AR, Probe	760 ± 77	910 ± 78

Table 3: Three-player limit Texas Hold'em winnings in mb/h and their 68% confidence intervals. Each match-up was run for at least 30000 hands. Player positions were permuted to balance positional advantages.

Conclusion

This work evaluates computer poker policies that have been trained from self-play MCTS. The introduced variant of UCT, Smooth IS-UCT, is shown to converge to a Nash equilibrium in a small poker game. Furthermore, it achieves robust performance in large limit Texas Hold'em games. This suggests that the policies it learns are close to a Nash equilibrium and with increased training time might also converge in large games. However, the results were not conclusive in demonstrating a higher performance of Smooth IS-UCT than plain IS-UCT, as both UCT-based methods achieved strong performance.

The results against FellOmen2 suggest that self-play MCTS can be competitive with the state of the art of 2008, a year in which algorithmic agents beat expert human players for the first time in a public competition. However, testing performance against benchmark opponents is only a partial assessment of the quality of a policy. It does not explicitly measure the potential exploitability and is therefore insufficient to assess a policy's distance to a Nash equilibrium.

We think that the following directions of research could improve the performance and understanding of self-play MCTS in computer poker. A study of the theoretical properties of Smooth IS-UCT might unveil a class of algorithms that are theoretically sound in partially observable environments and converge to Nash equilibria. Obtaining results from longer training periods and comparing the learning rate against state of the art methods, e.g. MCCFR, should help understanding the role of self-play MCTS in imperfect information games. A more sophisticated abstraction technique, e.g. discretisation through clustering or distribution-aware methods (Johanson et al. 2013), should improve the performance in Texas Hold'em. Using local or online search to plan for the currently played scenario could scale to multi-player games with more than three players. Due to their fast initial learning rate, this might be a particularly suitable application of MCTS-based methods.

Acknowledgments

This research was supported by the UK PhD Centre in Financial Computing and Google DeepMind.

References

Auer, P.; Cesa-Bianchi, N.; Freund, Y.; and Schapire, R. E. 1995. Gambling in a rigged casino: The adversarial multi-armed bandit problem. In *Foundations of Computer Science, 1995. Proceedings., 36th Annual Symposium on*, 322–331. IEEE.

Auer, P.; Cesa-Bianchi, N.; and Fischer, P. 2002. Finite-time analysis of the multiarmed bandit problem. *Machine learning* 47(2-3):235–256.

Auger, D. 2011. Multiple tree for partially observable monte-carlo tree search. In *Applications of Evolutionary Computation*. Springer. 53–62.

Billings, D.; Burch, N.; Davidson, A.; Holte, R.; Schaeffer, J.; Schauenberg, T.; and Szafron, D. 2003. Approximating game-theoretic optimal strategies for full-scale poker. In *IJCAI*, 661–668.

Billings, D. 2006. *Algorithms and assessment in computer poker*. Ph.D. Dissertation, University of Alberta.

Coulom, R. 2006. Efficient selectivity and backup operators in monte-carlo tree search. In *5th International Conference on Computer and Games*.

Cowling, P. I.; Powley, E. J.; and Whitehouse, D. 2012. Information set monte carlo tree search. *Computational Intelligence and AI in Games, IEEE Transactions on* 4(2):120–143.

Ganzfried, S., and Sandholm, T. 2008. Computing an approximate jam/fold equilibrium for 3-player no-limit texas hold'em tournaments. In *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems-Volume 2*, 919–925.

Gelly, S.; Kocsis, L.; Schoenauer, M.; Sebag, M.; Silver, D.; Szepesvári, C.; and Teytaud, O. 2012. The grand challenge of computer go: Monte carlo tree search and extensions. *Communications of the ACM* 55(3):106–113.

Gilpin, A., and Sandholm, T. 2006. A competitive texas hold'em poker player via automated abstraction and real-time equilibrium computation. In *Proceedings of the National Conference on Artificial Intelligence*, volume 21, 1007.

Hoda, S.; Gilpin, A.; Pena, J.; and Sandholm, T. 2010. Smoothing techniques for computing nash equilibria of sequential games. *Mathematics of Operations Research* 35(2):494–512.

Johanson, M.; Burch, N.; Valenzano, R.; and Bowling, M. 2013. Evaluating state-space abstractions in extensive-form games. In *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*, 271–278.

Kocsis, L., and Szepesvári, C. 2006. Bandit based monte-carlo planning. In *Machine Learning: ECML 2006*. Springer. 282–293.

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

Lanctot, M.; Waugh, K.; Zinkevich, M.; and Bowling, M. H. 2009. Monte carlo sampling for regret minimization in extensive games. In *NIPS*, 1078–1086.

Leslie, D. S., and Collins, E. J. 2006. Generalised weakened fictitious play. *Games and Economic Behavior* 56(2):285–298.

Littman, M. L. 1996. *Algorithms for sequential decision making*. Ph.D. Dissertation, Brown University.

Ponsen, M.; de Jong, S.; and Lanctot, M. 2011. Computing approximate nash equilibria and robust best-responses using sampling. *Journal of Artificial Intelligence Research* 42(1):575–605.

Risk, N. A., and Szafron, D. 2010. Using counterfactual regret minimization to create competitive multiplayer poker agents. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: volume 1-Volume 1*, 159–166.

Rubin, J., and Watson, I. 2011. Computer poker: A review. *Artificial Intelligence* 175(5):958–987.

Sandholm, T. 2010. The state of solving large incomplete-information games, and application to poker. *AI Magazine* 31(4):13–32.

Selten, R. 1975. Reexamination of the perfectness concept for equilibrium points in extensive games. *International journal of game theory* 4(1):25–55.

Silver, D., and Veness, J. 2010. Monte-carlo planning in large pomdps. In *Advances in Neural Information Processing Systems*, 2164–2172.

Silver, D.; Sutton, R. S.; and Müller, M. 2012. Temporal-difference search in computer go. *Machine learning* 87(2):183–219.

Tesauro, G. 1992. Practical issues in temporal difference learning. In *Reinforcement Learning*. Springer. 33–53.

Waugh, K.; Schnizlein, D.; Bowling, M.; and Szafron, D. 2009. Abstraction pathologies in extensive games. In *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems-Volume 2*, 781–788.

Zinkevich, M.; Johanson, M.; Bowling, M.; and Piccione, C. 2007. Regret minimization in games with incomplete information. In *Advances in neural information processing systems*, 1729–1736.