Approximating Optimal Dudo Play with Fixed-Strategy Iteration Counterfactual Regret Minimization

Todd W. Neller and Steven Hnath

Gettysburg College, Dept. of Computer Science, Gettysburg, Pennsylvania, 17325, USA,
tneller@gettysburg.edu, steve.hnath@gmail.com,
http://cs.gettysburg.edu/~tneller

Abstract. Using the bluffing dice game Dudo as a challenge domain, we abstract information sets using imperfect recall of actions. Even with such abstraction, the standard Counterfactual Regret Minimization (CFR) algorithm proves impractical for Dudo, with the number of recursive visits to the same abstracted information sets increasing exponentially with the depth of the game graph. By holding strategies fixed across each training iteration, we show how CFR training iterations may be transformed from an exponential-time recursive algorithm into a polynomial-time dynamic-programming algorithm, making computation of an approximate Nash equilibrium for the full 2-player game of Dudo possible for the first time.

1 Introduction

In recent years, Counterfactual Regret Minimization (CFR) has proven an important innovation in advancing optimal strategy approximation for large extensive game-trees of partially-observable stochastic games (POSGs) such as Texas Hold'em Poker [1–3]. Imperfect recall of actions is one means of abstracting such large extensive games, and the approximation of optimal play with this abstraction has led to play performance improvements with the non-abstracted game [4]. In this paper, we demonstrate how a simple modification to CFR using imperfect recall of actions reduces individual training iterations from exponential to polynomial time complexity, allowing the first computation of approximately optimal strategy for the full game of 2-player Dudo. After introducing Dudo, we will review imperfect recall of actions and CFR, motivate and introduce Fixed-Strategy Iteration CFR (FSICFR), and compare the performance of CFR and FSICFR. Finally, we will discuss open questions.

1.1 The Game of Dudo

Dudo is a bluffing dice game thought to originate from the Inca Empire circa 15$^{\text{th}}$ century. Many variations exist in both folk and commercial forms. The ruleset we use from [5] is perhaps the simplest representative form, and is thus most easily accessible to both players and researchers. Liar's Dice, Bluff, Call My Bluff, Perudo, Cacho, Cachito are names of variations[1].

[1] In some cases, e.g. Liar's Dice and Cacho, there are different games of the same name.

Dudo has been a popular game through the centuries. From the Inca Empire, Dudo spread to a number of Latin American countries, and is thought to have come to Europe via Spanish conquistadors [6]. It is said to have been "big in London in the 18th century" [7]. Richard Borg's commercial variant, published under the names Call My Bluff, Bluff, and Liar's Dice, won the prestigious Spiel des Jahres (German Game of the Year) in 1993. On BoardGameGeek.com[2], the largest website for board game enthusiasts, Liar's Dice is ranked 270/53298 (i.e. top 0.5%)[3]. Although a single, standard form of the game has not emerged, there is strong evidence of the persistence of the core game mechanics of this favorite bluffing dice game since its creation.

Rules: Each player is seated around a table and begins with five standard six-sided dice and a dice cup. Dice are lost with the play of each round, and the object of the game is to be the last player remaining with dice. At the beginning of each round, all players simultaneously roll their dice once, and carefully view their rolled dice while keeping them concealed from other players. The starting player makes a claim about what the players have *collectively* rolled, and players clockwise in turn each either make a stronger claim or challenge the previous claim, declaring "Dudo" (Spanish for "I doubt it."). A challenge ends the round, players lift their cups, and one of the two players involved in the challenge loses dice. Lost dice are placed in full view of players.

Claims consist of a positive number of dice and a rank of those dice, e.g. two 5's, seven 3's, or two 1's. In Dudo, the rank of 1 is *wild*, meaning that dice rolls of rank 1 are counted in totals for other ranks as well. We will denote a claim of n dice of rank r as $n \times r$. In general, one claim is stronger than another claim if there is an increase in rank and/or number of dice. That is, a claim of 2×4 may, for example, be followed by 2×6 (increase in rank) or 4×3 (increase in number). The exception to this general rule concerns claims of wild rank 1. Since 1's count for other ranks and other ranks do not count for 1's, 1's as a rank occur with half frequency in counts and are thus considered doubly strong in claims. So in the claim ordering, 1×1, 2×1, and 3×1 immediately precede 2×2, 4×2, and 6×2, respectively.

Mathematically, one may enumerate the claims in order of strength by defining $s(n, r)$, the strength of claim $n \times r$, as follows:

$$s(n,r) = \begin{cases} 5n - \lfloor \frac{n}{2} \rfloor - r - 7 & \text{if } r \neq 1 \\ 11n - 6 & \text{if } r = 1 \text{ and } r \leq \lfloor \frac{d_{\text{total}}}{2} \rfloor \\ 5d_{\text{total}} + n - 1 & \text{if } r = 1 \text{ and } r > \lfloor \frac{d_{\text{total}}}{2} \rfloor \end{cases} \quad (1)$$

where d_{total} is the total number of dice in play. Thus for 2 players with 1 die each, the claims would be numbered:

Strength $s(n,r)$	0	1	2	3	4	5	6	7	8	9	10	11
Claim $n \times r$	1×2	1×3	1×4	1×5	1×6	1×1	2×2	2×3	2×4	2×5	2×6	2×1

Play proceeds clockwise from the round-starting player with claims of strictly increasing strength until one player challenges the previous claimant with "Dudo". At

[2] `http://www.boardgamegeek.com`

[3] as of August 17th, 2011

this point, all cups are lifted, dice of the claimed rank (including wilds) are counted and compared against the claim. For example, suppose that Ann, Bob and Cal are playing Dudo, and Cal challenges Bob's claim of 7×6. There are three possible outcomes:

- **The actual rank count exceeds the challenged claim.** In this case, the challenger loses a number of dice equal to the difference between the actual rank count and the claim count. Example: Counting 6's and 1's, the actual count is 10. Thus, as an incorrect challenger, Cal loses $10 - 7 = 3$ dice.
- **The actual rank count is less than the challenged claim.** In this case, the challenged player loses a number of dice equal to the difference between the claim count and the actual rank count. Example: Counting 6's and 1's, the actual count is 5. Thus, as a correctly challenged claimant, Bob loses $7 - 5 = 2$ dice.
- **The actual rank count is equal to the challenged claim.** In this case, every player except the challenged player loses a single die. Example: Counting 6's and 1's, the actual count is indeed 7 as Bob claimed. In this special case, Ann and Cal lose 1 die each to reward Bob's exact claim.

In the first round, an arbitrary player makes the first claim. The winner of a challenge makes the first claim of the subsequent round. When a player loses all remaining dice, the player loses and exits the game. The last remaining player is the winner. The following table provides a transcript of an example 2-player game with "1:" and "2:" indicating information relevant to each player:

Round	Actions	Revealed Rolls		Result
1	1:"2×6", 2:"3×6", 1:"5×6", 2:"Dudo"	1:12566	2:23556	1:loses 1 die
2	2:"3×6", 1:"4×5", 2:"5×5", 1:"Dudo"	1:3555	2:23455	1:loses 1 die
3	2:"3×6", 1:"Dudo"	1:356	2:24466	1:loses 1 die
4	2:"2×2", 1:"3×2", 2:"Dudo"	1:12	2:13456	2:loses 1 die
5	1:"2×6", 2:"3×2", 2:"Dudo"	1:26	2:1222	1:loses 2 dice

2 Imperfect Recall and Counterfactual Regret Minimization

In this work, we restrict our attention to two-player Dudo, yet even with this simplification, we will show that the number of information sets poses difficulties for modern computing.

Since Dudo is divided into rounds, the play environment is episodic in nature and information from previous rounds is not relevant for the decision at hand[4]. An *information set* consists of sequences of moves and chance outcomes consistent with a player's state of knowledge. Thus, a Dudo information set consists of (1) a history of claims from the current round, (2) the player's private roll information, and (3) the number of opponent dice. Let d_1, d_2, and $d_{\max}$ denote the number of current player 1 dice, the number of opponent player 2 dice, and the maximum number of dice per player, respectively. In general, the number of information sets for d_1 and d_2 in a two-player game is

[4] The focus here is on optimal play. When seeking rather to model and exploit a suboptimal opponent, play information from previous rounds would, of course, be relevant.

then the product of the number of possible claim histories $2^{6(d_1+d_2)}$ (i.e. the size of the power set of possible claims), and the number of possible rolls $\binom{d_1+5}{d_1}$.

However, in Dudo it is impossible to have $d_1 < d_2 = d_{\max}$ with an even number of claims. An even number of claims in a claim history implies that the current player started the current round, and thus either the current round is the first round, or the current player won the previous round-ending challenge. However, $d_1 < d_{\max}$ implies this is not the first round, and $d_2 = d_{\max}$ implies the opponent could not have lost a previous challenge. Therefore, the current player lost the previous challenge, leading us to a contradiction. Since exactly half of power sets have an even-numbered size, we must reduce such information set counts by half. A symmetric argument applies for $d_2 < d_1 = d_{\max}$ with an odd number of claims. Our computation counting the number of power sets then becomes

$$\sum_{d_1,d_2 \in [1,5]} \begin{cases} 2^{6(d_1+d_2)-1}\binom{d_1+5}{d_1} & \text{if } d_1 < d_2 = d_{\max} \text{ or } d_2 < d_1 = d_{\max}, \\ 2^{6(d_1+d_2)}\binom{d_1+5}{d_1} & \text{otherwise.} \end{cases}$$
$$= 294,021,177,291,188,232,192.$$

Thus, a full 2-player, 5-versus-5 dice game of Dudo has over 294 quintillion information sets.

2.1 Imperfect Recall of Actions

For common modern machines, 2.9×10^{20} information sets is too large to iterate over for convergence of mixed strategies. As with successful computational approaches to Texas Hold'em Poker, we abstract information sets in order to reduce the problem size. We then solve the abstraction and apply the abstracted policy to the original game. For Dudo, the growth rate of possible claim sequences is most responsible for the overall growth of the extensive game-tree. We also note that the later claims of a round are less-easily supported and thus more often contain reliable information.

Since more recent claims tend to be more important to the decision at hand, our chosen means of abstraction is to form abstract information sets that recall up to m previous claims. For example, consider this 5-vs.-5 round claim sequence: 1×5, 2×5, 4×2, 5×4, 3×1, 6×4, 7×2. For a claim memory limit of $m = 3$, the 5-vs.-5 round information set for each of these decision would be enumerated according to the current player dice roll enumeration and the enumeration of the up-to-3 most recent claims: $\{\}, \{1 \times 5\}, \{1 \times 5; 2 \times 5\}, \{1 \times 5; 2 \times 5; 4 \times 2\}, \ldots, \{3 \times 1; 6 \times 4; 7 \times 2\}$.

Imperfect recall of actions allows us to trade off fine distinction of judgment for computational space and time requirements. Figure 1 shows how the number of abstract information sets varies according to the memory limit m and the number of dice for each player. As we will see, we can apply imperfect recall of actions without significantly impacting play performance.

2.2 Counterfactual Regret Minimization

Counterfactual Regret Minimization (Algorithm 1), while defined for general extensive game-trees, can also be applied to abstracted information sets. We will now summarize

m	**Information Sets**
1	57626
2	2069336
3	21828536
4	380033636
5	2751474854
all	2.9×10^{20}

(a) No. of abstracted info. sets with varying memory limit m.

	Opponent Dice				
Dice	**1**	**2**	**3**	**4**	**5**
1	1794	5928	13950	27156	43056
2	20748	48825	95046	163947	241962
3	130200	253456	437192	693504	971264
4	570276	983682	1560384	2327598	3132108
5	159012	217224	284508	360864	9084852

(b) Information sets for each round with memory limit $m = 3$.

Fig. 1. Number of abstracted information sets varying recall limit and number of dice in round.

the Counterfactual Regret Minimization (CFR) algorithm, directing the reader to [1] and [2] for detailed descriptions and proofs. At each player node recursively visited in a training iteration, a mixed strategy is computed according to the regret-matching equation, for which we now provide notation and define in a manner similar to [2].

Let A denote the set of all game *actions*. Let I denote an *information set*, and $A(I)$ denote the set of legal actions for information set I. Let t and T denote time steps. (Within both algorithms, t is with respect to each information set and is incremented with each visit to the information set.) A *strategy* σ_i^t for player i maps each player i information set I_i and legal player i action $a \in A(I_i)$ to the probability that the player will choose a in I_i at time t. All player strategies together at time t form a *strategy profile* σ^t. We refer to a strategy profile that excludes player i's strategy as σ_{-i}. Let $\sigma_{I \to a}$ denote a strategy equivalent to σ, except that action a is always chosen in information set I.

Let $\pi^\sigma(h)$ be the reach probability of game history h with strategy profile σ. Further, let $\pi^\sigma(I)$ be the reach probability of reaching information set I through all possible game histories in I, i.e. $\pi^\sigma(I) = \sum_{h \in I} \pi^\sigma(h)$. The counterfactual reach probability of information state I, $\pi^\sigma_{-i}(I)$, is the probability of reaching I with strategy profile σ except that, counter to the fact of σ we treat current player i actions to reach the state as having probability 1. In all situation we refer to as "counterfactual", one treats the computation as if player i's strategy was modified to have intentionally played to information set I_i. Put another way, we *exclude* the probabilities that factually came into player i's play from the computation.

Let Z denote the set of all terminal game histories. Then proper prefix $h \sqsubset z$ for $z \in Z$ is a nonterminal game history. Let $u_i(z)$ denote the utility to player i of terminal history z. Define the *counterfactual value* at nonterminal history h as:

$$v_i(\sigma, h) = \sum_{z \in Z, h \sqsubset z} \pi^\sigma_{-i}(h) \pi^\sigma(h, z) u_i(z). \tag{2}$$

The *counterfactual regret* of not taking a at history h is defined as:

$$r(h, a) = v_i(\sigma_{I \to a}, h) - v_i(\sigma, h). \tag{3}$$

The *counterfactual regret* of not taking action a at information set I is then:

$$r(I,a) = \sum_{h \in I} r(h,a) \tag{4}$$

The difference between the value of always choosing action a and the average value of the node strategy is an action's regret, which is then weighted by the probability that other player(s) will play to reach the node. This is then averaged over all time steps. If we define the nonnegative counterfactual regret $r_i^{T,+}(I,a) = \max(r_i^T(I,a), 0)$, then Hart and Mas-Colell's regret-matching equation [8] for the strategy at time $T+1$ is:

$$\sigma_i^{T+1}(I,a) = \begin{cases} \frac{r_i^{T,+}(I,a)}{\sum_{a \in A(I)} r_i^{T,+}(I,a)} & \text{if } \sum_{a \in A(I)} r_i^{T,+}(I,a) > 0 \\ \frac{1}{|A(I)|} & \text{otherwise.} \end{cases} \tag{5}$$

For each player node, this equation is used to compute action probabilities in proportion to the positive average regrets. For each action, CFR then computes associated child nodes for each player, and computes utilities of such actions through recursive calls with updated probabilities for reaching such nodes through the current node. Regrets are computed from the returned values, and the value of playing to the current node is finally computed and returned.

The CFR algorithm is presented in detail in Algorithm 1. The parameters to CFR are the history of actions, the learning player, the time step, and the reach probabilities for players 1 and 2, respectively. Variables beginning with v are for local computation and are not computed according to the previous equations for counterfactual value. In lines 13, 15, and 20, $P(h)$ is the active player after history h. In lines 14 and 16, ha denotes history h with appended action a. In line 22, π_{-i} refers to the counterfactual reach probability of the node, which in the case of players $\{1,2\}$ is the same as reach probability π_{3-i}. In line 32, $\emptyset$ refers to the empty history.

The average strategy profile at information set I, $\bar{\sigma}^T$, approaches an equilibrium as $T \to \infty$. The average strategy at information set I, $\bar{\sigma}^T(I)$, is obtained by normalizing s_I over all actions $a \in A(I)$. What is most often misunderstood about CFR is that this *average* strategy profile, and not the final strategy profile, is what converges to a Nash equilibrium.

3 Fixed-Strategy Iteration CFR

The technique we introduce in this paper, Fixed-Strategy Iteration Counterfactual Regret Minimization (FSICFR), is a significant structural modification to chance-sampled CFR yet convergence for both relies on the regret-matching equation (5) which is common to both algorithms.

Essentially, CFR traverses extensive game subtrees, recursing forward with reach probabilities that each player will play to each node (i.e. information set) while maintaining history, and backpropagating values and utilities used to update parent node action regrets and thus future strategy.

When applying chance-sampled CFR to abstracted Dudo, we observed that the number of recursive visits to abstracted player nodes grew exponentially with the depth

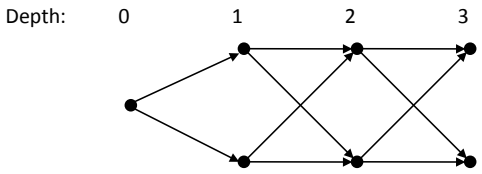
Algorithm 1 Counterfactual Regret Minimization

Initialize cumulative regret tables: $\forall I, r_I[a] \leftarrow 0$.
Initialize cumulative strategy tables: $\forall I, s_I[a] \leftarrow 0$.
Initialize initial profile: $\sigma^1(I, a) \leftarrow 1/|A(I)|$

function CFR(h, i, t, π_1, π_2):
if h is terminal **then**
  return $u_i(h)$
end if
Let I be the information set containing h.
$v_\sigma \leftarrow 0$
$v_{\sigma_{I \to a}}[a] \leftarrow 0$ for all $a \in A(I)$
for $a \in A(I)$ **do**
  if $P(h) = 1$ **then**
    $v_{\sigma_{I \to a}}[a] \leftarrow$ CFR(ha, i, t, $\sigma^t(I, a) \cdot \pi_1$, π_2)
  else if $P(h) = 2$ **then**
    $v_{\sigma_{I \to a}}[a] \leftarrow$ CFR(ha, i, t, π_1, $\sigma^t(I, a) \cdot \pi_2$)
  end if
  $v_\sigma \leftarrow v_\sigma + \sigma^t(I, a) \cdot v_{\sigma_{I \to a}}[a]$
end for
if $P(h) = i$ **then**
  for $a \in A(I)$ **do**
    $r_I[a] \leftarrow r_I[a] + \pi_{-i} \cdot (v_{\sigma_{I \to a}}[a] - v_\sigma)$
    $s_I[a] \leftarrow s_I[a] + \pi_i \cdot \sigma^t(I, a)$
  end for
  $\sigma^{t+1}(I) \leftarrow$ regret-matching values computed using Equation 5 and regret table r_I
end if
return v_σ

function Solve():
for $t = \{1, 2, 3, \ldots, T\}$ **do**
  for $i \in \{1, 2\}$ **do**
    CFR($\emptyset$, i, t, 1, 1)
  end for
end for

of the tree. Consider Figure 2, a simple directed acyclic graph (DAG). With increasing depth, the number of possible paths to a node grows exponentially. If we continue the DAG pattern to greater depth, we have a linear growth of nodes and exponential growth of paths.



Depth	Paths
1	1
2	2
3	4

Fig. 2. Example directed acyclic graph.

Similarly, in the game of Dudo with imperfect recall of actions, there are many paths to an abstracted information set. (For the remainder of the paper, assume that information sets are abstracted information sets, and that player nodes represent such abstracted information sets.) Each added die linearly grows the number of possible claims, which linearly grows the depth of the game-DAG and exponentially grows the number of paths to each information set. As will be seen in the experimental results, this exponential growth makes the application of CFR to the full game of Dudo impractical.

Fixed-Strategy Iteration CFR (FSICFR) divides the recursive CFR algorithm into two iterative passes, one forward and one backward, through a DAG of nodes. On the forward pass, visit counts and reach probabilities of each player are accumulated, yet all strategies remain fixed. (By contrast, in CFR, the strategy at a node is updated with each CFR visit.) After all visits are counted and probabilities are accumulated, a backward pass computes utilities and updates regrets. FSICFR computational time complexity is proportional to the number of *nodes* times the average node outdegree, whereas CFR complexity is proportional to the number of *node visits* times the average node outdegree. Since the number of node visits grows exponentially in such abstracted problems, FSICFR has exponential savings in computational time per training iteration relative to CFR.

We now present the FSICFR algorithm for two-player, zero-sum games in detail as Algorithm 2. Each player node n consists of a number of fields:

$visits$ - the sum of possible paths to this node during a single training iteration.
$pSum_i$ - the sum of the probabilities that player i would play to this node along each possible path during a single training iteration.
r - a mapping from each possible action to the average counterfactual regret for not choosing that action.
σ - a node strategy, i.e. a probability distribution function mapping each possible action to the probability of choosing that action.
σSum - the sum of the strategies used over each training iteration the node is visited.
$player$ - the player currently taking action.
T - the sum of $visits$ across all training iterations.

v - the expected game value for the current player at that node.

A significant, domain-specific assumption is required for this simple, zero-sum form of FSICFR: There is a one-to-one correspondence between player nodes and abstracted information sets, and our visits to nodes must contain enough state information (e.g. predetermined public and private information for both players) such that the appropriate successor nodes may be chosen. In the case of Dudo, this means that the algorithm, having predetermined player rolls, knows which player information sets are legal successors.

We note that the predetermination of chance node outcomes may present difficulties for some games or game abstractions where constraints on a chance node are dependent on the path by which it is reached (e.g. drawing the next card without knowing how many and thus which cards have been previously drawn). This does not pose a problem for Dudo or non-draw forms of Poker. Observe that such predetermination should proceed in topological order, as predetermination of chance nodes affects reachability of later chance nodes.

Also, we note that if, as in the case of Dudo, the number of visits to a node will be constant across all training iterations, then one can eliminate the $n.visits$ variable and replace it with the value 1 in equations.

In summary, consider FSICFR as being similar to the case where we hold the strategy before a CFR training iteration fixed and execute an entire iteration of regret updates without changing strategy until after the iteration. In such a case, all operations at a node are the same, and we transform the exponential tree-recursion of CFR into a linear dynamic-programming graph-traversal for FSICFR. As a consequence, node update frequency is equalized rather than exponentially proportional to depth.

4 Experimental Results

Not only did the performance of FSICFR exceed that of CFR for small numbers of dice in play, but it was also *necessary* to the approximation of optimal strategy for the full 5-versus-5 game of Dudo. A single iteration of standard CFR became prohibitively expensive for practical Dudo training for even a small number of dice. In this section, we will begin with experiments to demonstrate real-time learning performance for the small 2-versus-2 game, compare iteration costs for both algorithms as we scale to more dice, and see how imperfect recall of actions performs when recall is varied[5].

4.1 Learning Outperformance

In order to understand the relative real-time strength of FSICFR versus CFR, we suspend the training threads at specific intervals of seconds (1, 2, 4, 8, etc.), exporting the current strategy and resuming training. CFR defines the strategy for unvisited player nodes as the default uniform strategy.

[5] All experiments in this section were performed on Dell Optiplex 990 computers with an Intel Core i7-2600 (3.4 GHz) and 8GB RAM running Ubuntu 11.04.

Algorithm 2 FSICFR(L)

Require: A topologically sorted list L of extensive game DAG nodes.
Ensure: The normalized strategy sum $n.\sigma Sum$, i.e. the average strategy for each player node n, approximates a Nash equilibrium.

for each training iteration **do**
  Predetermine the chance outcomes at all reachable chance nodes in L.
  for each node n in order of topologically-sorted, reachable subset $L' \subset L$ **do**
    if n is the initial node **then**
      $n.visits \leftarrow n.pSum_1 \leftarrow n.pSum_2 \leftarrow 1$
    end if
    if n is a player node **then**
      Compute strategy $n.\sigma$ according to Equation 5.
      if $n.r$ has no positive components **then**
        Let $n.\sigma$ be the uniform distribution.
      end if
      $n.\sigma Sum \leftarrow n.\sigma Sum + (n.\sigma \cdot n.pSum_1$ **if** $n.player = 1$ **else** $n.\sigma \cdot n.pSum_2)$
      for each action a **do**
        Let c be the associated child of taking action a in n with probability $n.\sigma(a)$.
        $c.visits \leftarrow c.visits + n.visits$
        $c.pSum_1 \leftarrow c.pSum_1 + (n.\sigma(a) \cdot n.pSum_1$ **if** $n.player = 1$ **else** $n.pSum_1)$
        $c.pSum_2 \leftarrow c.pSum_2 + (n.pSum_2$ **if** $n.player = 1$ **else** $n.\sigma(a) \cdot n.pSum_2)$
      end for
    else if n is a chance node **then**
      Let c be the associated child of the predetermined chance outcome for n.
      $c.visits \leftarrow c.visits + n.visits$
      $c.pSum_1 \leftarrow c.pSum_1 + n.pSum_1$
      $c.pSum_2 \leftarrow c.pSum_2 + n.pSum_2$
    end if
  end for
  for each node n in reverse order of topologically-sorted, reachable subset $L' \subset L$ **do**
    if n is a player node **then**
      $n.v \leftarrow 0$
      for each action a **do**
        Let c be the associated child of taking action a in n with probability $n.\sigma(a)$.
        $n.v(a) \leftarrow (c.v$ **if** $n.player = c.player$ **else** $-c.v)$
        $n.v \leftarrow n.v + n.\sigma(a) \cdot n.v(a)$
      end for
      Counterfactual probability $cfp \leftarrow (n.pSum_2$ **if** $n.player = 1$ **else** $n.pSum_1)$
      for each action a **do**
        $n.r(a) \leftarrow \frac{1}{n.T + n.visits}(n.T \cdot n.r(a) + n.visits \cdot cfp \cdot (n.v(a) - n.v))$
      end for
      $n.T \leftarrow n.T + n.visits$
    else if n is a chance node **then**
      Let $[n.player, n.v] \leftarrow [c.player, c.v]$, where c is the predetermined child of n.
    else if n is a terminal node **then**
      $n.v \leftarrow$ the utility of n for current player $n.player$.
    end if
    $n.visits \leftarrow n.pSum_1 \leftarrow n.pSum_2 \leftarrow 0$
  end for
end for

With a claim history memory limit of 3, we first sought to compare performance for 1-vs.-1 die training, comparing learned strategy versus optimal strategy. However, within the first second, both algorithms had largely converged, yielding little contrast. We therefore turned our attention to 2-vs.-2 die training. Such training relies upon pre-computed results of 1-vs.-2 and 2-vs.-1 dice training, both of which rely upon 1-vs.-1 die training. For fair comparison, we used the result of 2 million iterations of FSICFR for these 2- and 3-dice strategies, giving both the same smaller-case training. For each exported strategy, we played 1 million games divided into two identical sets of 500K games with the two players trading positions for each set.

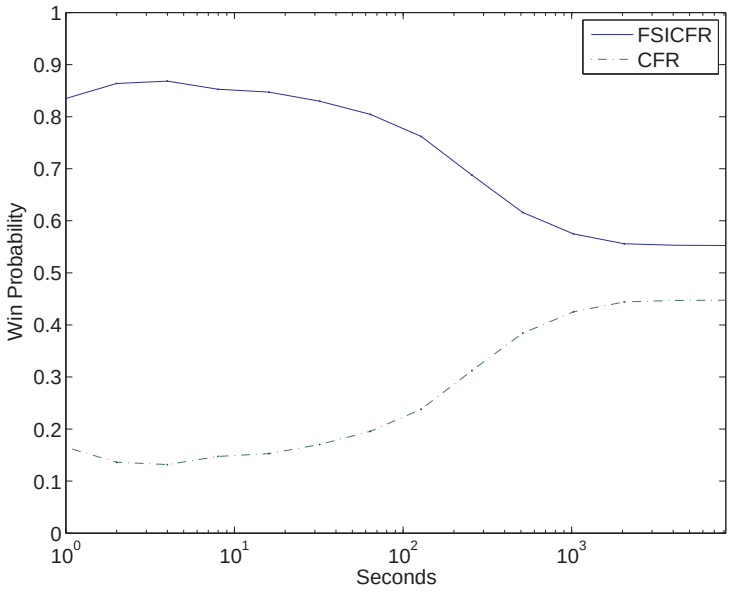


Fig. 3. FSICFR vs. CFR win rates during 2-vs.-2 dice training.

The observed win rates with respect to logarithmic seconds are shown in Figure 3. The contrast in the relative speeds of the two algorithms was so great that CFR training still had unvisited nodes and made some use of the default random strategy through 128 seconds whereas FSICFR training had visited all nodes before the first export at 1 second. We can see that FSICFR starts with a significant advantage, peaking at an 86.8% win rate at 4 seconds, after which CFR training gradually decreases that edge. Since both algorithms rely on the same update equations, they eventually converge to the same game value. However, CFR's convergence is slow even for 2-vs.-2 dice and, after 8192 seconds of training, still wins only 44.7% of games against FSICFR[6].

[6] Wilson score intervals were used to compute 90% confidence intervals, but such intervals are so small for 1 million games that they are not easily seen in Figure 3.

4.2 Computational Time Per Training Iteration

The full recursive traversal of CFR causes the relatively slow computation of training iterations and underperformance against FSICFR. Furthermore, when we increase the problem size, FSICFR becomes completely necessary. Consider the time cost per training iteration of FSICFR (Figure 4(a)) and CFR (Figure 4(b)). Entries are omitted in Figure 4(b) where a single training iteration for a single pair of die rolls takes more than four days of computation[7].

	Opponent Dice				
Dice	**1**	**2**	**3**	**4**	**5**
1	0.3	1.0	2.1	4.9	9.9
2	0.7	2.0	4.7	9.7	18.7
3	2.1	4.7	9.2	18.6	34.4
4	5.0	9.8	18.7	32.4	55.3
5	10.0	18.9	34.8	56.1	94.5

(a) Time (ms) per FSICFR training iteration averaged over 1000 training iterations.

	Opponent Dice				
Dice	**1**	**2**	**3**	**4**	**5**
1	13	49	1551	97210	6179541
2	26	1507	99019	6310784	-
3	1509	98483	6211111	-	-
4	97611	6265326	-	-	-
5	6290658	-	-	-	-

(b) Time (ms) per single CFR training iteration.

Fig. 4. Time (ms) per training iteration.

Whereas the time cost per FSICFR iteration roughly doubles with each added die in play, the standard CFR cost increases by almost two orders of magnitude for each added die. Thus, for more dice in play, FSICFR not only outperforms CFR, but also converges for all combinations of player/opponent dice before CFR can complete a *single* training iteration for a *single* simulated pair of dice rolls with 7 or more dice. Computation of Dudo full-game strategy is not feasible with standard CFR. FSICFR makes feasible not only Dudo strategy computation, but also strategy computation for similarly structured games with exponentially growing paths to nodes of greater depth.

4.3 Varying Imperfect Action Recall

Finally, we turn our attention to the real-time training performance of FSICFR when we vary the imperfect action recall limit. As in earlier experiments, we suspend training threads and export strategies at intervals. In this experiment, however, we train for the simplest 1-vs.-1 die round and compare performance against a known optimal 1-vs.-1 die strategy computed by Todd Neller and Megan Knauss in 2007 using the technique of [9]. The results are shown in Figure 5.

First, we observe that by 512 seconds all training for memories of 2 or more claims converges within 0.15% of optimal performance. Only limiting recall to a single claim shows poor, erratic performance. Smaller claim memory limits yield smaller extensive game-trees and converge more quickly with generally lesser performance. However,

[7] E.g., a single iteration of 2-vs.-5 dice CFR required $389244980\text{ms} \approx 4.5\text{days}$.

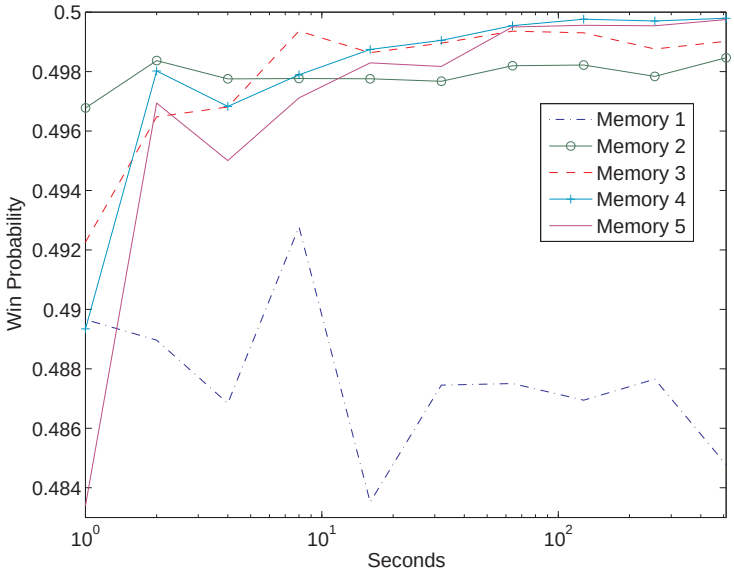


Fig. 5. 1-vs.-1 die FSICFR win rates vs. optimal strategy varying action recall imperfection.

the eventual superior gain in performance is not so significant for larger claim memory limits. Thus, a claim memory limit of 3 appears to be a reasonable abstraction for high-quality Dudo play that approximates optimal strategy.

Finally, we note that neither CFR nor FSICFR are guaranteed to converge to a Nash equilibrium when applied to imperfect recall abstractions. Such abstraction pathologies have been studied in [10]. However, experimental evidence (e.g. Figure 5) indicates no such pathological behavior for the application of FSICFR to our imperfect recall abstraction of Dudo.

5 Conclusion

In this work, we have computed the first approximation of optimal play for the enduringly-popular 15$^{\text{th}}$-century Inca dice game of Dudo. Abstracting the game through imperfect recall of actions, we reduced the number of information sets from over 2.9×10^{20} to under 2.2×10^7. However, Counterfactual Regret Minimization (CFR), a powerful recent technique for competitive computer Texas Hold'em Poker, proved impractical for application due to exponentially growing recursive path traversals in even the abstracted game.

Our primary contribution is the observation that, in such games, the exponential recursive traversal of CFR can be restructured as a polynomial dynamic-programming traversal if we hold strategies fixed at each node across the entire forward traversal, updating the regrets and strategies only once per node at the concluding backpropagation of utilities. Fixed-Strategy Iteration Counterfactual Regret Minimization (FSICFR)

proved not only to be superior in learning rate for a small number of dice in play, but also necessary for the practical computation of an approximate Nash equilibrium for the full 2-player game starting with 5-vs.-5 dice.

We conclude with a number of open questions: What is the true game value for the full game of Dudo, and what is the quality of our full-game approximation? What is the best parallel version of this algorithm? Given an approximation of optimal policy, what is the best way to compress the policy so as to minimize the loss of performance? While we have taken significant steps forward, we recognize that many interesting questions lie ahead.

The authors would like to thank Marc Lanctot for his great assistance in presenting the overview of CFR, and Megan Knauss for her contribution to the first computation of optimal 1-vs.-1 die Dudo strategy.

References

1. Zinkevich, M., Johanson, M., Bowling, M., Piccione, C.: Regret minimization in games with incomplete information. In Platt, J., Koller, D., Singer, Y., Roweis, S., eds.: Advances in Neural Information Processing Systems 20. MIT Press, Cambridge, MA (2008) 1729–1736
2. Lanctot, M., Waugh, K., Zinkevich, M., Bowling, M.: Monte carlo sampling for regret minimization in extensive games. In Bengio, Y., Schuurmans, D., Lafferty, J., Williams, C.K.I., Culotta, A., eds.: Advances in Neural Information Processing Systems 22. MIT Press (2009) 1078–1086
3. Risk, N.A., Szafron, D.: 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. AAMAS '10, Richland, SC, International Foundation for Autonomous Agents and Multiagent Systems (2010) 159–166
4. Waugh, K., Zinkevich, M., Johanson, M., Kan, M., Schnizlein, D., Bowling, M.H.: A practical use of imperfect recall. In Bulitko, V., Beck, J.C., eds.: SARA, AAAI (2009)
5. Knizia, R.: Dice Games Properly Explained. Elliot Right-Way Books, Brighton Road, Lower Kingswood, Tadworth, Surrey, KT20 6TD U.K. (1999)
6. Mohr, M.S.: The New Games Treasury. Houghton Mifflin, Boston, Massachusetts (1993)
7. Jacobs, G.: The World's Best Dice Games, new edition. John N. Hansen Co., Inc., Milbrae, California (1993)
8. Hart, S., Mas-Colell, A.: A simple adaptive procedure leading to correlated equilibrium. Econometrica **68**(5) (September 2000) 1127–1150
9. Koller, D., Megiddo, N., von Stengel, B.: Fast algorithms for finding randomized strategies in game trees. In: Proceedings of the 26th ACM Symposium on Theory of Computing (STOC '94). (1994) 750–759
10. Waugh, K., Schnizlein, D., Bowling, M.H., Szafron, D.: Abstraction pathologies in extensive games. In Sierra, C., Castelfranchi, C., Decker, K.S., Sichman, J.S., eds.: AAMAS (2), IFAAMAS (2009) 781–788