Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence

Solving Imperfect Information Games Using Decomposition

Neil Burch, Michael Johanson and Michael Bowling
Computing Science Department, University of Alberta
{nburch,johanson,mbowling}@ualberta.ca

Abstract

Decomposition, *i.e.,* independently analyzing possible subgames, has proven to be an essential principle for effective decision-making in perfect information games. However, in imperfect information games, decomposition has proven to be problematic. To date, all proposed techniques for decomposition in imperfect information games have abandoned theoretical guarantees. This work presents the first technique for decomposing an imperfect information game into subgames that can be solved independently, while retaining optimality guarantees on the full-game solution. We can use this technique to construct theoretically justified algorithms that make better use of information available at run-time, overcome memory or disk limitations at run-time, or make a time/space trade-off to overcome memory or disk limitations while solving a game. In particular, we present an algorithm for subgame solving which guarantees performance in the whole game, in contrast to existing methods which may have unbounded error. In addition, we present an offline game solving algorithm, CFR-D, which can produce a Nash equilibrium for a game that is larger than available storage.

Introduction

A game solving algorithm takes the description of a game and computes or approximates an optimal strategy (*i.e.,* a Nash equilibrium) for playing the game. Perfect information games, such as checkers, where game states are entirely public, have historically been more tractable to solve than imperfect information games, such as poker, where some information about the game state is hidden from one or more players. The main reason is that perfect information games can easily be partitioned into subgames that can be solved independently, producing strategy fragments that can be combined to form an optimal strategy for the entire game.

Reasoning about subgames independently has two highly desirable properties. First, decomposition can allow large savings in the memory required to solve a game. If we split a game with S states into subgames half-way to the end of the game, we end up with $\mathcal{O}(S)$ subgames each of size $\mathcal{O}(\sqrt{S})$: a single "trunk" spanning from the start of the game to the split depth, plus a number of subgames. If we only need to reason about a single subgame at a time, then we use an amount of storage on the order of $\mathcal{O}(\sqrt{S})$ instead of $\mathcal{O}(S)$. The subgame pieces can also be recursively decomposed, so that in perfect information games that are no more than D actions long, a game solving algorithm like depth-first iterative-deepening (Korf 1985) uses only $\mathcal{O}(D)$ memory. Second, we do not need to store the complete strategy, which may be too large to store, but rather can recompute the subgame strategies as needed. As a result such perfect information decomposition algorithms are effectively not limited by space, and with sufficient time can solve extremely large games. For example, checkers with 5×10^{20} states has been solved (Schaeffer et al. 2007) both in terms of the game's value and an optimal Nash equilibrium strategy.

In imperfect information games, there are currently no methods for solving, or re-solving, subgames with a guarantee that the subgame strategies can be combined into an equilibrium for the whole game. State-of-the-art algorithms are all limited to comparatively small problems where the complete strategy fits in available space. As a result, 2-Player Limit Texas Hold'em Poker, with 9×10^{17} game states, is smaller than checkers but has not been solved despite considerable recent interest in the game. Computing an optimal strategy for this game would require hundreds of terabytes of memory using a state-of-the-art game solving algorithm.

In this paper we present, for the first time, two methods which safely use decomposition in imperfect information games. We give a new definition of subgames which is useful for imperfect information games, and a method for re-solving these subgames which is guaranteed to not increase the exploitability (*i.e.,* suboptimality) of a strategy for the whole game. We also give a general method called CFR-D for computing an error-bounded approximation of a Nash equilibrium through decomposing and independently analyzing subgames of an imperfect information game. Finally, we give experimental results comparing our new methods to existing techniques, showing that the prior lack of theoretical bounds can lead to significant error in practice.

Notation and Background

An extensive-form game is a model of sequential interaction of one or more agents or players. Let P be the set of players. Let H be the set of all possible game states, represented as the history of actions taken from the initial game state $\varnothing$. The state $h \cdot a \in H$ is a child of the state h, h is the parent of $h \cdot a$, and h' is a descendant of h or $h \sqsubset h'$ if h is any strict prefix of h'. Let Z be the set of all terminal states. For each non-terminal state h, $A(h)$ gives the set of legal actions, and $P(h) \in P \cup \{c\}$ gives the player to act, where c denotes the "chance player", which represents stochastic events outside of the players' control. $\sigma_c(h, a)$ is the probability that chance will take action $a \in A(h)$ from state h, and is common knowledge. H_p is the set of all states h such that $P(h) = p$. For every $z \in Z$, $u_p(z) \in \Re$ gives the payoff for player p if the game ends in state z. If $p = \{1, 2\}$ and $u_1(z) + u_2(z) = 0$ for all $z \in Z$, we say the game is two-player, zero-sum.

The information structure of the game is described by information sets for each player p, which form a partition $\mathcal{I}_p$ of H_p. For any information set $I \in \mathcal{I}_p$, any two states $h, j \in I$ are indistinguishable to player p. Let $I(h)$ be the information set in $\mathcal{I}_p$ which contains h. A behaviour strategy $\sigma_p \in \mathbf{\Sigma}_p$ is a function $\sigma_p(I, a) \in \Re$ which defines a probability distribution over valid actions for every information set $I \in \mathcal{I}_p$. We will say $\sigma_p(h, a) = \sigma_p(I(h), a)$, since a player cannot act differently depending on information they did not observe. Let $Z(I) = \{z \in Z \text{ s.t. } z \sqsupset h \in I\}$ be the set of all terminal states z reachable from some state in information set I. We can also consider the terminal states reachable from I after some action a, stated as $Z(I, a) = \{z \in Z \text{ s.t. } z \sqsupset h \cdot a, h \in I\}$.

In games with perfect recall, any two states h and j in an information set $I \in \mathcal{I}_p$ have the same sequence of player p information sets and actions. Informally, perfect recall means that a player does not forget their own actions or any information observed before making those actions. As a result, for any $z \in Z(I)$ there is a unique state $h \in I$ such that $h \sqsubset z$, which we write $z[I]$. This paper focuses exclusively on two player, zero-sum, perfect recall games.

A strategy profile $\sigma \in \mathbf{\Sigma}$ is a tuple of strategies, one for each player. Given σ, it is useful to refer to certain products of probabilities. Let $\pi^\sigma(h) = \prod_{j \cdot a \sqsubseteq h} \sigma_{P(j)}(j, a)$, which gives the joint probability of reaching h if all players follow σ. We use $\pi_p^\sigma(h)$ to refer to the product of only the terms where $P(h) = p$, and $\pi_{-p}^\sigma(h)$ to refer to the product of terms where $P(h) \neq p$. Note that in games with perfect recall, for all states h, h' in $I \in \mathcal{I}_p$, $\pi_p(h) = \pi_p(h')$, so we can also speak of $\pi_p(I)$. We use $\pi^\sigma(j, h)$ to refer to the product of terms from j to h, rather than from $\varnothing$ to h. If we replace the whole strategy for player p by a new strategy σ'_p, we will call the resulting profile $\langle \sigma_{-p}, \sigma'_p \rangle$. Finally, $\sigma_{[S \leftarrow \sigma']}$ is the strategy that is equal to σ everywhere except at information sets in S, where it is equal to σ'.

Given a strategy profile σ, the expected utility u_p^σ to player p if all players follow σ is $\sum_Z \pi^\sigma(z) u_p(z)$. The expected utility $u_p^\sigma(I, a)$ of taking an action at an information set is $\sum_{z \in Z(I,a)} \pi^\sigma(z) u_p(z)$. In this paper, we will frequently use a variant of this expected value called counterfactual value: $v_p^\sigma(I, a) = \sum_{z \in Z(I,a)} \pi_{-p}^\sigma(z) \pi_p^\sigma(z[I] \cdot a, z) u_p(z)$. Informally, the counterfactual value of I for player p is the expected value of reaching I if p plays to reach I.

A best response $\mathrm{BR}_p(\sigma) = \mathrm{argmax}_{\sigma'_p \in \mathbf{\Sigma}_p} u_p^{\langle \sigma_{-p}, \sigma'_p \rangle}$ is a strategy for p which maximises p's value if all other player strategies remain fixed. A Nash equilibrium is a strategy profile where all strategies are simultaneously best responses to each other, and an ϵ-Nash equilibrium is a profile where the expected value for each player is within ϵ of the value of a best response strategy. In two-player, zero-sum games, the expected utility of any Nash equilibrium is a game-specific constant, called the game value. In a two-player zero-sum game, we use the term exploitability to refer to a profile's average loss to a best response across its component strategies. A Nash equilibrium has an exploitability of zero.

A counterfactual best response $\mathrm{CBR}_p(\sigma)$ is a strategy where $\sigma_p(I, a) > 0$ if and only if $v_p(I, a) \geq \max_b v_p(I, b)$, so it maximizes counterfactual value at every information set. CBR_p is necessarily a best response, but BR_p may not be a counterfactual best response as it may choose non-maximizing actions where $\pi_p(I) = 0$. The well known recursive bottom-up technique of constructing a best response generates a counterfactual best response.

Decomposition into Subgames

In this paper we introduce a new refinement on the concept of a subgame. A subgame, in a perfect information game, is a tree rooted at some arbitrary state: a set of states closed under the descendant relation. The state-rooted subgame definition is not as useful in an imperfect information game because the tree cuts across information set boundaries: for any state s in the tree, there is generally at least one state $t \in I(s)$ which is not in the tree.

To state our refined notion of subgame it is convenient to extend the concept of an information set. $I(h)$ is defined in terms of the states which player $p = P(h)$ cannot distinguish. We would also like to partition states where player p acts into those which player $p' \neq p$ cannot distinguish. We use the ancestor information sets to construct $I_{p'}(h)$, the augmented information set for player p' containing h. Let $H_{p'}(h)$ be the sequence of player p' information sets reached by player p' on the path to h, and the actions taken by player p'. Then for two states h and j, $I_{p'}(h) = I_{p'}(j) \iff H_{p'}(h) = H_{p'}(j)$.

We can now state the following definition of a subgame:

Definition 1 *An imperfect information subgame is a forest of trees, closed under both the descendant relation and membership within augmented information sets for any player.*

The imperfect information subgame is a forest rooted at a set of augmented information sets. If state s is in the subgame, and $s \sqsubseteq t$ or $s, t \in I_p$ for any information set I_p, then state t is also in the subgame. Note that the root of the subgame will not generally be a single information set, because different players will group states into different information sets. We use augmented information sets in this definition because we wish to preserve the information partitioning at the root of the subgame. For example, say player one can distinguish states s and t where player one is acting, and player two can distinguish their descendants, but not their ancestors. If we did not use augmented information sets, a subgame could include s and not include t. We use augmented information sets to rule out this case. In a

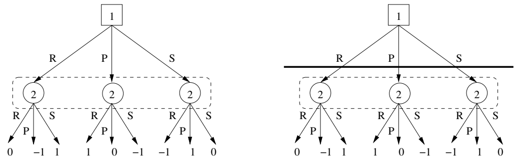

Figure 1: Left: rock-paper-scissors. Right: rock-paper-scissors split into trunk and one subgame.

perfect information game, our definition is equivalent to the usual definition of a subgame, as information sets all contain a single state.

We will use the game of rock-paper-scissors as a running example in this paper. In rock-paper-scissors, two players simultaneously choose rock, paper, or scissors. They then reveal their choice, with rock beating scissors, scissors beating paper, and paper beating rock. The simultaneous moves in rock-paper-scissors can be modeled using an extensive form game where one player goes first, without revealing their action, then the second player acts. The extensive form game is shown on the left side of Figure 1. The dashed box indicates the information set $I_2 = \{R, P, S\}$ which tells us player two does not know player one's action.

On the right side of Figure 1, we have decomposed the game into two parts: a trunk containing state $\varnothing$ and a single subgame containing three states R, P, and S. In the subgame, there is one player two information set $I_2 = \{R, P, S\}$ and three augmented player one information sets $I_1^R = \{R\}$, $I_1^P = \{P\}$, and $I_1^S = \{S\}$.

Subgame Strategy Re-Solving

In this section we present a method of re-solving a subgame, using some compact summary information retained from a previous strategy in this subgame. The novel property of this method is a bound on the exploitability of the combined trunk and new subgame strategy in the whole game. This sort of re-solving problem might be useful in a number of situations. For example, we might wish to move a strategy from some large machine to one with very limited memory. If we can re-solve subgame strategies as needed, then we can discard the original subgame strategies to save space. Another application occurs if the existing strategy is suboptimal, and we wish to find a better subgame strategy with a guarantee that the new combined trunk and subgame strategy does at least as well as the existing strategy in the worst case. An additional application of space reduction while solving a game is presented later in this paper.

First, we note that it is not sufficient to simply re-solve the subgame with the assumption that the trunk policy is fixed. When combined with the trunk strategy, multiple subgame solutions may achieve the same expected value if the opponent can only change their strategy in the subgame, but only a subset of these subgame strategies will fare so well against a best response where the opponent can also change their strategy in the trunk.

Consider the rock-paper-scissors example. Let's say we started with an equilibrium, and then discarded the strategy in the subgame. In the trunk, player one picks uniformly between R, P, and S. In the subgame, player one has only one possible (vacuous) policy: they take no actions. To find an equilibrium in the subgame, player two must pick a strategy which is a best response to the empty player one policy, given the probability of $\frac{1}{3}$ for R, P, and S induced by the trunk strategy. All actions have an expected utility of 0, so player two can pick an arbitrary policy. For example, player two might choose to always play rock. Always playing rock achieves the game value of 0 against the combined trunk and subgame strategy for player one, but gets a value of -1 if player one switched to playing paper in the trunk.

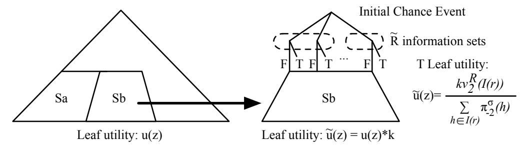

Figure 2: Construction of the Re-Solving Game

Our new method of re-solving subgames relies on summarising a subgame strategy with the opponent's counterfactual values $v_{opp}(I)$ for all information sets I at the root of the subgame. $v_{opp}(I)$ gives the "what-if" value of our opponent reaching the subgame through information set I, if they changed their strategy so that $\pi_{opp}(I) = 1$. In rock-paper-scissors, the player one counterfactual values for R, P, and S are all 0 in the equilibrium profile. When player two always played rock in the example above, the player one counterfactual values for R, P, and S were 0, 1, and -1 respectively. Because the counterfactual value for P was higher than the original equilibrium value of 0, player one had an incentive to switch to playing P. That is, they could change their trunk policy to convert a larger "what-if" counterfactual value into a higher expected utility by playing P.

If we generate a subgame strategy where the opponent's best response counterfactual values are no higher than the opponent's best response counterfactual values for the original strategy, then the exploitability of the combined trunk and subgame strategy is no higher than the original strategy. From here on, we will assume, without loss of generality, that we are re-solving a strategy for player 1.

Theorem 1 *Given a strategy σ_1, a subgame S, and a re-solved subgame strategy σ_1^S, let $\sigma_1' = \sigma_{1,[S \leftarrow \sigma_1^S]}$ be the combination of σ_1 and σ_1^S. If $v_2^{\langle \sigma_1', \mathrm{CBR}(\sigma_1') \rangle}(I) \leq v_2^{\langle \sigma_1, \mathrm{CBR}(\sigma_1) \rangle}(I)$ for all information sets I at the root of subgame S, then $u_2^{\langle \sigma_1', \mathrm{CBR}(\sigma_1') \rangle} \leq u_2^{\langle \sigma_1, \mathrm{CBR}(\sigma_1) \rangle}$.*

A proof of Theorem 1 is given in Burch *et al.* (Burch, Johanson, and Bowling 2013).

To re-solve for a strategy in a subgame, we will construct the modified subgame shown in Figure 2. We will distinguish the re-solving game from the original game by using a tilde ($\tilde{}$) to distinguish states, utilities, or strategies for the re-solving game. The basic construction is that each state r at the root of the original subgame turns into three states: a p_2 choice node $\tilde{r}$, a terminal state $\tilde{r} \cdot T$, and a state $\tilde{r} \cdot F$ which is identical to r. All other states in the original subgame are directly copied into the re-solving game. We must also be given $v_2^R(I) \equiv v_2^{\langle \sigma_1, \mathrm{CBR}(\sigma_1) \rangle}(I)$ and $\pi_{-2}^{\sigma}(I)$ for all p_2 information sets $I \in \mathcal{I}_2^R$ at the root of the subgame.

The re-solving game beings with an initial chance node which leads to states $\tilde{r} \in \tilde{R}$, corresponding to the probability of reaching state $r \in R$ in the original game. Each state $\tilde{r} \in \tilde{R}$ occurs with probability $\pi^{\sigma}_{-2}(r)/k$, where the constant $k = \sum_{r \in R} \pi^{\sigma}_{-2}(r)$ is used to ensure that the probabilities sum to 1. $\tilde{R}$ is partitioned into information sets $\mathcal{I}_2^{\tilde{R}}$ that are identical to the information sets $\mathcal{I}_2^R$.

At each $\tilde{r} \in \tilde{R}$, p_2 has a binary choice of F or T. After T, the game ends. After F, the game is the same as the original subgame. All leaf utilities are multiplied by k to undo the effects of normalising the initial chance event. So, if $\tilde{z}$ corresponds to a leaf z in the original subgame, $\tilde{u}_2(\tilde{z}) = ku_2(z)$. If $\tilde{z}$ is a terminal state after a T action, $\tilde{u}_2(\tilde{z}) = \tilde{u}_2(\tilde{r} \cdot T) = kv_2^R(I(r))/\sum_{h \in I(r)} \pi^{\sigma}_{-2}(h)$. This means that for any $I \in \mathcal{I}_2^{\tilde{R}}$, $\tilde{u}_2(I \cdot T) = v_2^R(I)$, the original counterfactual best response value of I.

No further construction is needed. If we solve the proposed game to get a new strategy profile $\tilde{\sigma^*}$, we can directly use $\tilde{\sigma^*}_1$ in the original subgame of the full game. To see that σ_1^* achieves the goal of not increasing the counterfactual values for p_2, consider $\tilde{u}_2(I)$ for $I \in \mathcal{I}_2^{\tilde{R}}$ in an equilibrium profile for the re-solving game. p_2 can always pick T at the initial choice to get the original counterfactual values, so $\tilde{u}_2(I) \geq v_2^R(I)$. Because v_2^R comes from $\langle \sigma_1, \mathrm{CBR}(\sigma_1)\rangle$, $\tilde{u}_2(I) \leq v_2^R(I)$ in an equilibrium. So, in a solution $\tilde{\sigma^*}$ to the re-solving game, $\tilde{u}_2(I) = v_2^R(I)$, and $\tilde{u}_2^{\langle \tilde{\sigma^*}_1, \mathrm{CBR}(\tilde{\sigma^*}_1)\rangle}(I \cdot F) \leq v_2^R(I)$. By construction of the re-solving game, this implies that $v_2^{\langle \sigma_1^*, \mathrm{CBR}(\sigma_1^*)\rangle}(I) \leq v_2^R(I)$.

If we re-solve the strategy for both players at a subgame, the exploitability of the combined strategy is increased by no more than $(|\mathcal{I}^{R_S}| - 1)\epsilon_S + \epsilon_R$, where ϵ_R is the exploitability of the subgame strategy in the re-solving subgame, ϵ_S is the exploitability of the original subgame strategy in the full game, and $|\mathcal{I}^{R_S}|$ is the number of information sets for both players at the root of a subgame. This is proven as Theorem 3 in Burch *et al.* (Burch, Johanson, and Bowling 2013).

Generating a Trunk Strategy using CFR-D

CFR-D is part of the family of counterfactual regret minimisation (CFR) algorithms, which are all efficient methods for finding an approximation of a Nash equilibrium in very large games. CFR is an iterated self play algorithm, where the average policy across all iterations approaches a Nash equilibrium (Zinkevich et al. 2008). It has independent regret minimisation problems being simultaneously updated at every information set, at each iteration. Each minimisation problem at an information set $I \in \mathcal{I}_p$ uses immediate counterfactual regret, which is just external regret over counterfactual values: $R^T(I) = \max_{a \in A(I)} \sum_t v^{\sigma^t}_{P(I)}(I, a) - \sum_{a'} \sigma^t(I, a')v^{\sigma^t}_{P(I)}(I, a')$. The immediate counterfactual regrets place an upper bound on the regret across all strategies, and an ϵ-regret strategy profile is a 2ϵ-Nash equilibrium (Zinkevich et al. 2008).

Using separate regret minimisation problems at each information set makes CFR a very flexible framework. First, any single regret minimisation problem at an information set I only uses the counterfactual values of the actions. The action probabilities of the strategy profile outside I are otherwise irrelevant. Second, while the strategy profile outside I is generated by the other minimisation problems in CFR, the source does not matter. Any sequence of strategy profiles will do, as long as they have low regret.

The CFR-BR algorithm (Johanson et al. 2012a) uses these properties, and provided the inspiration for the CFR-D algorithm. The game is split into a trunk and a number of subgames. At each iteration, CFR-BR uses the standard counterfactual regret minimisation update for both players in the trunk, and for one player in the subgames. For the other player, CFR-BR constructs and uses a best response to the current CFR player strategy in each subgame.

In our proposed algorithm, CFR-D, we use a counterfactual best response in each subgame for both players. That is, at each iteration, one subgame at a time, we solve the subgame given the current trunk strategy, update the trunk using the counterfactual values at the root of the subgame, update the average counterfactual values at the root of the subgame, and then discard the solution to the subgame. We then update the trunk using the current trunk strategy. The average strategy is an approximation of a Nash equilibrium, where we don't know any action probabilities in the subgames. Note that we must keep the average counterfactual values at the root of the subgames if we wish to use subgame re-solving to find a policy in the subgame after solving.

Theorem 2 *Let $\mathcal{I}_{TR}$ be the information sets in the trunk, $A = \max_{I \in \mathcal{I}} |A(I)|$ be an upper bound on the number of actions, and $\Delta = \max_{s,t \in Z} |u(s) - u(t)|$ be the variance in leaf utility. Let σ^t be the current CFR-D strategy profile at time t, and N_S be the number of information sets at the root of any subgame. If for all times t, players p, and information sets I at the root of a subgame $\mathcal{I}_{SG}$, the quantity $R^T_{full}(I) = \max_{\sigma'} v_p^{\sigma^t_{[\mathcal{I}_{SG} \leftarrow \sigma']}}(I) - v_p^{\sigma^t}(I)$ is bounded by ϵ_S, then player p regret $R_p^T \leq \Delta|\mathcal{I}_{TR}|\sqrt{AT} + TN_S\epsilon_S$.*

Proof The proof follows from Zinkevich *et al.*'s argument in Appendix A.1 (Zinkevich et al. 2008). Lemma 5 shows that for any player p information set I, $R^T_{full}(I) \leq R^T(I) + \sum_{I' \in Child_p(I)} R^T_{full}(I')$ where $Child_p(I)$ is the set of all player information sets which can be reached from I without passing through another player p information set.

We now use an argument by induction. For any trunk information set I with no descendants in $\mathcal{I}_{TR}$, we have $R^T_{full}(I) \leq R^T(I) \leq R^T(I) + TN_S\epsilon_S$.

Assume that for any player p information set I with no more than $i \geq 0$ descendants in $\mathcal{I}_{TR}$, $R^T_{full}(I) \leq \sum_{I' \in Trunk(I)} R^T(I') + TN_S\epsilon_S$, where $Trunk(I)$ is the set of player p information sets in $\mathcal{I}_{TR}$ reachable from I, including I. Now consider a player p information set with $i+1$ descendants. By Lemma 5 of Zinkevich *et al.*, we get $R^T_{full} \leq R^T(I) + \sum_{I' \in Child_p(I)} R^T_{full}(I')$. Because I' must have no more than i descendants for all $I' \in Child(I)$, we get $R^T_{full}(I) \leq \sum_{I' \in Trunk(I)} R^T(I') + TN_S\epsilon_S$.

By induction this holds for all i, and must hold at the root of the game, so $R_p^T \leq \sum_{I \in \mathcal{I}_{TR}} R^T(I') + TN_S\epsilon_S$. We do regret matching in the trunk, so $R^T(I') \leq \Delta\sqrt{AT}$ for all I'. □

The benefit of CFR-D is the reduced memory requirements. CFR-D only stores values for information sets in the trunk and at the root of each subgame, giving it memory requirements which are sub-linear in the number of information sets. Treating the subgames independently can lead to a substantial reduction in space: $\mathcal{O}(\sqrt{S})$ instead of $\mathcal{O}(S)$, as described in the introduction. There are two costs to the reduced space. The first is that the subgame strategies must be re-solved at run-time. The second cost is increased CPU time to solve the game. At each iteration, CFR-D must find a Nash equilibrium for a number of subgames. CFR variants require $\mathcal{O}(1/\epsilon^2)$ iterations to have an error less than ϵ, and this bound applies to the number of trunk iterations required for CFR-D. If we use CFR to solve the subgames, each of the subgames will also require $\mathcal{O}(1/\epsilon^2)$ iterations at each trunk iteration, so CFR-D ends up doing $\mathcal{O}(1/\epsilon^4)$ work.

In CFR-D, the subgame strategies must be mutual counterfactual best responses, not just mutual best responses. The only difference is that a counterfactual best response will maximise counterfactual value at an information set I where $\pi_{P(I)}(I) = 0$. A best response may choose an arbitrary policy at I. While CFR naturally produces a mutual counterfactual best response, a subgame equilibrium generated by some other method like a sequence form linear program may not be a counterfactual best response. In this case, the resulting strategy profile is easily fixed with a post-processing step which computes the best response using counterfactual values whenever $\pi_p^\sigma(I)$ is 0.

Experimental Results

We have three main claims to demonstrate. First, if we have a strategy, we can reduce space usage by keeping only summary information about the subgames, and then re-solve any subgame with arbitrarily small error. Second, we can decompose a game, only use space for the trunk and a single subgame, and generate an arbitrarily good approximation of a Nash equilibrium using CFR-D. Finally, we can use the subgame re-solving technique to reduce the exploitability of an existing strategy. All results were generated on a 2.67GHz Intel Xeon X5650 based machine running Linux.

Re-Solving Strategies in Subgames

To show that re-solving subgames introduces at most an arbitrarily small exploitability, we use the game of Leduc Hold'em poker, a popular research testbed for imperfect information games (Waugh et al. 2009; Ganzfried, Sandholm, and Waugh 2011). The game uses a 6-card deck and has two betting rounds, with 936 information sets total. It retains interesting strategic elements while being small enough that a range of experiments can be easily run and evaluated. In this experiment, the trunk used was the first round of betting, and there were five subgames corresponding to the five different betting sequences where no player folds. When re-solving for subgame strategies, we used the Public Chance Sampling (PCS) variant of CFR (Johanson et al. 2012b).

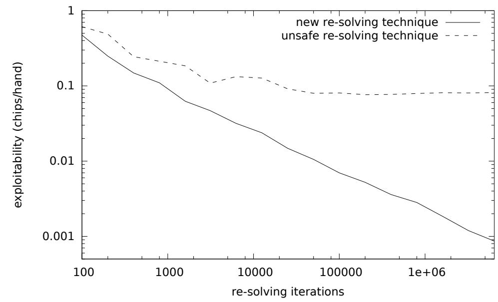

Figure 3: exploitability after subgame re-solving

To demonstrate the practicality of re-solving subgame strategies, we started with an almost exact Nash equilibrium (exploitable by less than $2.5 * 10^{-11}$ chips per hand), computed the counterfactual values of every hand in each subgame for both players, and discarded the strategy in all subgames. These steps correspond to a real scenario where we pre-compute and store a Nash equilibrium in an offline fashion. At run-time, we then re-solved each subgame using the subgame re-solving game constructed from the counterfactual values and trunk strategy, and measured the exploitability of the combined trunk and re-solved subgame strategies.

Figure 3 shows the exploitability when using a different number of CFR iterations to solve the re-solving games. The $\mathcal{O}(1/\sqrt{T})$ error bound for CFR in the re-solving games very clearly translates into the expected $\mathcal{O}(1/\sqrt{T})$ error in the overall exploitability of the re-constructed strategy.

For comparison, the "unsafe re-solving technique" line in Figure 3 shows the performance of a system for approximating undominated subgame solutions (Ganzfried and Sandholm 2013). Not only is there no theoretical bound on exploitability, the real world behaviour is not ideal. Instead of approaching a Nash equilibrium (0 exploitability), the exploitability of the re-solved strategy approaches a value of around 0.080 chips/hand. Re-solving time ranged from 1ms for 100 iterations, up to 25s for 6.4 million iterations, and the safe re-solving method was around one tenth of a percent slower than unsafe re-solving.

Solving Games with Decomposition

To demonstrate CFR-D, we split Leduc Hold'em in the same fashion as the strategy re-solving experiments. Our implementation of CFR-D used CFR for both solving subgames while learning the trunk strategy and the subgame re-solving games. All the reported results use 200,000 iterations for each of the re-solving subgames (0.8 seconds per subgame.) Each line of Figure 4 plots the exploitability for different numbers of subgame iterations performed during CFR-D, ranging from 100 to 12,800 iterations. There are results for 500, 2,000, 8,000, and 32,000 trunk iterations.

Looking from left to right, each of the lines show the decrease in exploitability as the quality of subgame solutions increases. The different lines compare exploitability across an increasing number of CFR-D iterations in the trunk.

Given that the error bound for CFR variants is $\mathcal{O}(\sqrt{T})$,

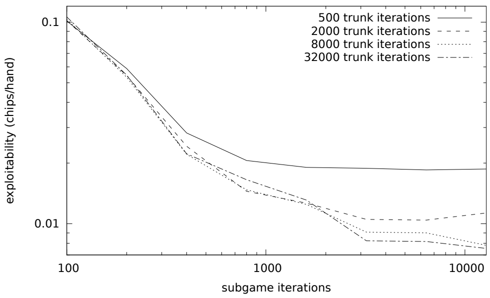

Figure 4: CFR-D exploitability

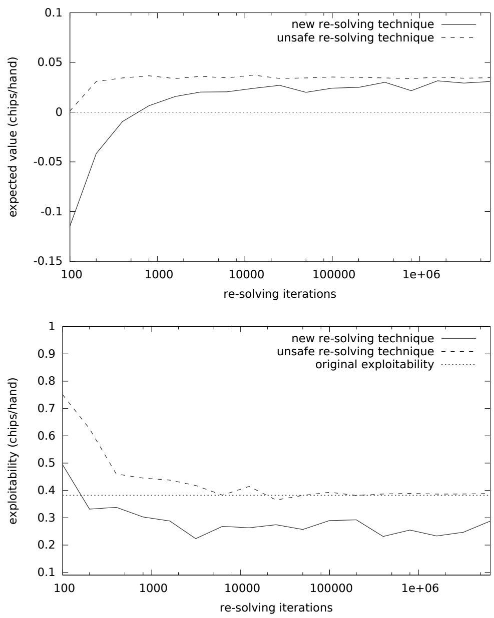

Figure 5: plots of re-solved abstract strategy in Leduc Hold'em showing performance against the original strategy (top) and exploitability (bottom)

one might expect exploitability results to be a straight line on a log-log plot. In these experiments, CFR-D is using CFR for the trunk, subgames, and the re-solving games, so the exploitability is a sum of trunk, subgame, and subgame re-solving errors. For each line on the graph, trunk and subgame re-solving error are constant values. Only subgame error decreases as the number of subgame iteration increases, so each line is approaching the non-zero trunk and re-solving error, which shows up as a plateau on a log-log plot.

Re-Solving to Improve Subgame Strategies

Generating a new subgame strategy at run-time can also be used to improve the exploitability of a strategy. In large games, lossy abstraction techniques are often used to reduce a game to a tractable size (Johanson et al. 2013). When describing their recent subgame solving technique, Ganzfried *et al.* reported positive results in experiments where subgames are re-solved using a much finer-grained abstract game than the original solution (Ganzfried and Sandholm 2013). Our new subgame re-solving method adds a theoretical guarantee to the ongoing research in this area.

In Figure 5, we demonstrate re-solving subgames with a Leduc Hold'em strategy generated using an abstraction. In the original strategy, the players can not tell the difference between a Queen or a King on the board if they hold a Jack, or between a Jack or a Queen on the board if they hold a King. This abstraction gives a player perfect knowledge of the strength of their hand against a uniform random hand, but loses strategically important "textural" information and the resulting strategy is exploitable for 0.382 chips/hand in the full game. To generate the counterfactual values needed for our method, we simply do a best response computation within the subgame: the standard recursive best response algorithm naturally produces counterfactual values.

The top plot shows the expected value of an abstract trunk strategy with re-solved unabstracted subgames, when played in the full game against the original abstract strategy. With little effort, both re-solving techniques see some improvement against the original strategy. With more effort, the unsafe re-solving technique has a small edge of around 0.004 chips/hand over our re-solving technique.

The bottom plot measures the exploitability of the re-solved strategies. Within 200 iterations, our new re-solving method decreases the exploitability to 0.33 chips/hand. After 2,000 iterations, the exploitability ranges between 0.23 and 0.29 chips/hand. The unsafe method, after 6,250 iterations, stays at 0.39 chips/hand. Note that the unsafe method's apparent convergence to the original exploitability is a coincidence: in other situations the unsafe strategy can be less exploitable, or significantly more exploitable.

If we have reliable information about the opponent's trunk strategy, we might want to use the unsafe re-solving method for its slight advantage in one-on-one performance. Otherwise, the large difference in exploitability between the re-solving methods supports our safe re-solving method. This produces a robust strategy with a guarantee that the re-solved strategy does no worse than the original strategy.

Conclusions

In perfect information games, decomposing the problem into independent subgames is a simple and effective method which is used to greatly reduce the space and time requirements of algorithms. It has previously not been known how to decompose imperfect information domains without a loss of theoretical guarantees on solution quality. We present a method of using summary information about a subgame strategy to generate a new strategy which is no more exploitable than the original strategy. Previous methods have no guarantees, and we demonstrate that they produce strategies which can be significantly exploitable in practice.

We also present CFR-D, an algorithm which uses decomposition to solve games. For the first time, we can use decomposition to achieve sub-linear space costs, at a cost of increased computation time. Using CFR-D, we can solve 2-Player Limit Texas Hold'em Poker in less than 16GB, even though storing a complete strategy would take over 200TB of space. While the time cost of solving Limit Hold'em is currently too large, this work overcomes one of the key barriers to such a computation being feasible.

Acknowledgements

This research was supported by the Natural Sciences and Engineering Research Council (NSERC), Alberta Innovates Centre for Machine Learning (AICML), and Alberta Innovates Technology Futures (AITF). Computing resources were provided by Compute Canada.

References

Burch, N.; Johanson, M.; and Bowling, M. 2013. Solving imperfect information games using decomposition. *CoRR* abs/1303.4441.

Ganzfried, S., and Sandholm, T. 2013. Improving performance in imperfect-information games with large state and action spaces by solving endgames. In *Procedings of the Computer Poker and Imperfect Information Workshop at AAAI 2013*.

Ganzfried, S.; Sandholm, T.; and Waugh, K. 2011. Strategy purification. In *Applied Adversarial Reasoning and Risk Modeling*.

Johanson, M.; Bard, N.; Burch, N.; and Bowling, M. 2012a. Finding optimal abstract strategies in extensive-form games. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence*, AAAI '12.

Johanson, M.; Bard, N.; Lanctot, M.; Gibson, R.; and Bowling, M. 2012b. Efficient nash equilibrium approximation through monte carlo counterfactual regret minimization. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems - Volume 2*, AAMAS '12, 837–846. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems.

Johanson, M.; Burch, N.; Valenzano, R.; and Bowling, M. 2013. Evaluating state-space abstractions in extensive-form games. In *Proceedings of the Twelfth International Conference on Autonomous Agents and Multiagent Systems (AAMAS-13)*.

Korf, R. E. 1985. Depth-first iterative-deepening: An optimal admissible tree search. *Artificial Intelligence* 27(1):97–109.

Schaeffer, J.; Burch, N.; Björnsson, Y.; Kishimoto, A.; Müller, M.; Lake, R.; Lu, P.; and Sutphen, S. 2007. Checkers is solved. *Science* 317(5844):1518.

Waugh, K.; Schnizlein, D.; Bowling, M.; and Szafron, D. 2009. Abstraction pathologies in extensive games. In *Proceedings of the Eighth International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, 781–788.

Zinkevich, M.; Johanson, M.; Bowling, M.; and Piccione, C. 2008. Regret minimization in games with incomplete information. In *Advances in Neural Information Processing Systems 20 (NIPS)*, 905–912.