Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence (IJCAI 2015)

Simultaneous Abstraction and Equilibrium Finding in Games

Noam Brown
Computer Science Department
Carnegie Mellon University
noamb@cs.cmu.edu

Tuomas Sandholm
Computer Science Department
Carnegie Mellon University
sandholm@cs.cmu.edu

Abstract

A key challenge in solving extensive-form games is dealing with large, or even infinite, action spaces. In games of imperfect information, the leading approach is to find a Nash equilibrium in a smaller abstract version of the game that includes only a few actions at each decision point, and then map the solution back to the original game. However, it is difficult to know which actions should be included in the abstraction without first solving the game, and it is infeasible to solve the game without first abstracting it.

We introduce a method that combines abstraction with equilibrium finding by enabling actions to be added to the abstraction at run time. This allows an agent to begin learning with a coarse abstraction, and then to strategically insert actions at points that the strategy computed in the current abstraction deems important. The algorithm can quickly add actions to the abstraction while provably not having to restart the equilibrium finding. It enables anytime convergence to a Nash equilibrium of the full game even in infinite games. Experiments show it can outperform fixed abstractions at every stage of the run: early on it improves as quickly as equilibrium finding in coarse abstractions, and later it converges to a better solution than does equilibrium finding in fine-grained abstractions.

1 Introduction

A central challenge in solving imperfect-information games is that the game may be far too large to solve with an equilibrium-finding algorithm. For example, two-player No-Limit Texas Hold'em poker, a popular game among humans and a leading testbed for research on solving imperfect-information games, has more than 10^{165} nodes in the game tree and 10^{161} information sets [Johanson, 2013]. For such games, *abstraction* has emerged as a key approach: a smaller, more tractable version of the game that maintains many of its strategic features is created [Shi and Littman, 2002; Billings *et al.*, 2003; Gilpin and Sandholm, 2007; 2006; Lanctot *et al.*, 2012; Johanson *et al.*, 2013; Kroer and Sandholm, 2014a; Ganzfried and Sandholm, 2014; Brown *et al.*, 2015]. In the abstract game, a player is constrained to behaving identically in a set of situations. The abstract game is then solved for (near-) equilibrium, and its solution (i.e., the strategies for all players) mapped back to the full game. Intuitively, an abstraction should retain "important" parts of the game as fine-grained as possible, while strategically similar or unimportant states could be grouped together. However, the key problem is that it is difficult to determine which states should be abstracted without knowing the equilibrium of the game. This begets a chicken-and-egg problem.

One approach is to solve for equilibrium in one abstraction, then use the equilibrium strategies to guide the generation of the next abstraction, and so on. Such *strategy-based abstraction* has been applied by going around that loop twice in Texas Hold'em, and within each iteration of the loop manually setting the abstraction (for the first betting round of the game) based on the equilibrium that was found computationally in the previous abstraction [Sandholm, 2010].

Another issue is that even if an equilibrium-finding algorithm in a given abstraction could determine what parts of the game are (un)important, the equilibrium-finding algorithm would have to be run from scratch on the new abstraction.

A related issue is that a given abstraction is good in the equilibrium-finding process only for some time. Coarse abstractions yield good strategies early in the run while larger, fine-grained abstractions take longer to reach reasonable strategies but yield better ones in the long run. Therefore, the abstraction size is typically hand crafted to the anticipated available run time of the equilibrium-finding algorithm using intuition and past experience with the approach.

There has been a great deal of research on generating good abstractions before equilibrium finding. Most of that work has focused on *information abstraction*, where a player is forced to ignore certain information in the game. Less research has gone into *action abstraction* (restricting the set of actions available to a player). Action abstraction has typically been done by hand using domain-specific knowledge (with some notable recent exceptions [Sandholm and Singh, 2012; Kroer and Sandholm, 2014a; 2014b]).

Beyond the manual strategy-based abstraction mentioned above, there has been a very limited amount of work on interleaving abstraction and equilibrium finding. Hawkin et al. [2011; 2012] proposed algorithms that adjust the sizes of some actions (some bet sizes in no-limit poker) in an abstrac-

tion during equilibrium finding, without convergence guarantees. Brown et al. [2014] presented an algorithm that adjusts action sizes during equilibrium finding in a way that guarantees convergence if the player's equilibrium value for the game is convex in the action-size vector. Those approaches seem to work for adjusting a small number of action sizes. A recent paper uses function approximation, which could be thought of as a form of abstraction, within equilibrium finding [Waugh *et al.*, 2015]. None of those approaches change the number of actions in the abstraction and thus cannot be used for growing (refining) an abstraction.

We present an algorithm that intertwines action abstraction and equilibrium finding. It begins with a coarse abstraction and selectively adds actions that the equilibrium-finding algorithm deems important. It overcomes the chicken-and-egg problem mentioned above and does not require knowledge of how long the equilibrium-finding algorithm will be allowed to run. It can quickly—in constant time—add actions to the abstraction while provably not having to restart the equilibrium finding. Experiments show it outperforms fixed abstractions at every stage of the run: early on it improves as quickly as equilibrium finding in coarse abstractions, and later it converges to a better solution than does equilibrium finding in fine-grained abstractions.

2 Setting, Notation, and Background

This section presents the notation we use. In an imperfect-information extensive-form game there is a finite set of players, P. H is the set of all possible histories (nodes) in the game tree, represented as a sequence of actions, and includes the empty history. $A(h)$ is the actions available in a history and $P(h) \in P \cup c$ is the player who acts at that history, where c denotes chance, which plays an action $a \in A(h)$ with a fixed probability $\sigma_c(h,a)$ that is known to all players. The history h' reached after an action is taken in h is a child of h, represented by $h \to a = h'$, while h is the parent of h'. More generally, h' is an ancestor of h, represented by $h' \sqsubset h$, if there exists a sequence of actions from h' to h. $Z \subseteq H$ are terminal histories for which no actions are available. For each player $i \in P$, there is a payoff function $u_i : Z \to \Re$. If $P = \{1,2\}$ and $u_1 = -u_2$, the game is a zero-sum game. We define $\Delta_i = \max_{z \in Z} u_i(z) - \min_{z \in Z} u_i(z)$ and $\Delta = \max_i \Delta_i$.

Imperfect information is represented by *information sets* for each player $i \in P$ by a partition $\mathcal{I}_i$ of $h \in H : P(h) = i$. For any information set $I \in \mathcal{I}_i$, all histories $h, h' \in I$ are indistinguishable to player i, so $A(h) = A(h')$. $I(h)$ is the information set I where $h \in I$. $P(I)$ is the player i such that $I \in \mathcal{I}_i$. $A(I)$ is the set of actions such that for all $h \in I$, $A(I) = A(h)$. $|A_i| = \max_{I \in \mathcal{I}_i} |A(I)|$ and $|A| = \max_i |A_i|$.

A strategy $\sigma_i(I)$ is a probability vector over $A(I)$ for player i in information set I. The probability of a particular action a is denoted by $\sigma_i(I,a)$. Since all histories in an information set belonging to player i are indistinguishable, the strategies in each of them must be identical. For all $h \in I$, $\sigma_i(h) = \sigma_i(I)$ and $\sigma_i(h,a) = \sigma_i(I,a)$. We define σ_i as a probability vector for player i over all available strategies Σ_i in the game. A strategy profile σ is a tuple of strategies, one for each player. $u_i(\sigma_i, \sigma_{-i})$ is the expected payoff for player i if all players play according to the strategy profile $\langle \sigma_i, \sigma_{-i} \rangle$. If a series of strategies are played over T iterations, then $\bar{\sigma}_i^T = \frac{\sum_{t \in T} \sigma_i^t}{T}$

$\pi^\sigma(h) = \Pi_{h' \to a \sqsubseteq h} \sigma_{P(h)}(h,a)$ is the joint probability of reaching h if all players play according to σ. $\pi_i^\sigma(h)$ is the contribution of player i to this probability (that is, the probability of reaching h if all players other than i, and chance, always chose actions leading to h). $\pi_{-i}^\sigma(h)$ is the contribution of all players other than i, and chance. $\pi^\sigma(h,h')$ is the probability of reaching h' given that h has been reached, and 0 if $h \not\sqsubset h'$. In a *perfect-recall* game, which we limit our discussion to in this paper, $\forall h, h' \in I \in \mathcal{I}_i$, $\pi_i(h) = \pi_i(h')$. Therefore, for $i = P(I)$ we define $\pi_i(I) = \pi_i(h)$ for $h \in I$. We define the average strategy to be $\bar{\sigma}_i^T(I) = \frac{\sum_{t \in T} \pi_i^{\sigma_i^t} \sigma_i^t(I)}{\sum_{t \in T} \pi_i^{\sigma^t}(I)}$.

In perfect-information games, a *subgame* is a subtree—rooted at some history—of the game tree. For convenience, we define an *imperfect information subgame (IISG)*. If a history is in an IISG, then any other history with which it shares an information set must also be in the IISG. Moreover, any descendent of the history must be in the IISG. Formally, an IISG is a set of histories $S \subseteq H$ such that for all $h \in S$, if $h \sqsubset h'$, then $h' \in S$, and for all $h \in S$, if $h' \in I(h)$ for some $I \in \mathcal{I}_{P(h)}$ then $h' \in S$. The *head* of an IISG S_r is the union of information sets that have actions leading directly into S, but are not in S. Formally, S_r is a set of histories such that for all $h \in S_r$, either $h \notin S$ and $\exists a \in A(h)$ such that $h \to a \in S$, or $h \in I$ and for some history $h' \in I$, $h' \in S_r$. We define the *greater subgame* $S^* = S \cup S_r$.

For equilibrium finding we use *counterfactual regret minimization (CFR)*, a regret-minimization algorithm for extensive-form games [Zinkevich *et al.*, 2007]. Recent research that finds close connections between CFR and other iterative learning algorithms [Waugh and Bagnell, 2015] suggests that our techniques may extend beyond CFR as well.

Our analysis of CFR makes frequent use of *counterfactual value*. Informally, this is the expected utility of an information set given that player i tries to reach it. For player i at information set I given a strategy profile σ, this is defined as

$$v_i^\sigma(I) = \sum_{h \in I} \left(\pi_{-i}^\sigma(h) \sum_{z \in Z} \left(\pi^\sigma(h,z) u_i(z) \right) \right) \tag{1}$$

The counterfactual value of an action a is

$$v_i^\sigma(I,a) = \sum_{h \in I} \left(\pi_{-i}^\sigma(h) \sum_{z \in Z} \left(\pi^\sigma(h \to a, z) u_i(z) \right) \right) \tag{2}$$

Let σ^t be the strategy profile used on iteration t. The *instantaneous regret* on iteration t for action a in information set I is $r_t(I,a) = v_{P(I)}^{\sigma^t}(I,a) - v_{P(I)}^{\sigma^t}(I)$. The *regret* for action a in I on iteration T is

$$R^T(I,a) = \sum_{t \in T} r_t(I,a) \tag{3}$$

Additionally, $R_+^T(I,a) = \max\{R^T(I,a), 0\}$ and $R^T(I) = \max_a\{R_+^T(I,a)\}$. Regret for player i in the entire game is

$$R_i^T = \max_{\sigma_i' \in \Sigma_i} \sum_{t \in T} u_i(\sigma_i', \sigma_{-i}^t) - u_i(\sigma_i^t, \sigma_{-i}^t) \tag{4}$$

In CFR, a player in an information set picks an action among the actions with positive regret in proportion to those regrets (and uniformly if all regrets are nonpositive). If a player plays according to CFR on every iteration, then on iteration T, $R^T(I) \leq \Delta_i \sqrt{|A(I)|}\sqrt{T}$. Moreover, $R_i^T \leq \sum_{I \in \mathcal{I}_i} R^T(I) \leq |\mathcal{I}_i| \Delta_i \sqrt{|A_i|}\sqrt{T}$. So, as $T \to \infty$, $\frac{R_i^T}{T} \to 0$.

In two-player zero-sum games, CFR converges to a *Nash equilibrium*, i.e., a strategy profile σ^* such that $\forall i$, $u_i(\sigma_i^*, \sigma_{-i}^*) = \max_{\sigma_i' \in \Sigma_i} u_i(\sigma_i', \sigma_{-i}^*)$. An *$\epsilon$-equilibrium* is a strategy profile σ^* such that $\forall i$, $u_i(\sigma_i^*, \sigma_{-i}^*) + \epsilon \geq \max_{\sigma_i' \in \Sigma_i} u_i(\sigma_i', \sigma_{-i}^*)$. If both players' average regrets $\frac{R_i^T}{T} \leq \epsilon$, their average strategies $\langle \bar{\sigma}_1^T, \bar{\sigma}_2^T \rangle$ form a 2ϵ-equilibrium.

3 Adding Actions to an Abstraction

A central challenge with abstraction is its inflexibility to change during equilibrium finding. After many iterations of equilibrium finding, one may determine that certain actions left out of the abstraction are actually important. Adding to an abstraction has been a recurring topic of research [Waugh *et al.*, 2009a; Gibson, 2014]. Prior work by Burch et al. [2014] examined how to reconstruct a Nash equilibrium strategy for an IISG after equilibrium finding completed by storing only the counterfactual values of the information sets in the head of the IISG. Jackson [2014] presented a related approach that selectively refines IISGs already in the abstraction after equilibrium finding completed. However, no prior method has guaranteed convergence to a Nash equilibrium if equilibrium finding continues after actions are added to an abstraction. The algorithm we present in this section builds upon these prior approaches, but provides this key theoretical guarantee.

In this section we introduce a general approach for adding actions to an abstraction at run time, and prove that we still converge to a Nash equilibrium. We discuss specifically adding IISGs. That is, after some number T_0 of CFR iterations in a game Γ, we wish to add some actions leading to an IISG S, forming Γ'. A trivial way to do this is to simply restart equilibrium finding from scratch after the IISG is added. However, intuitively, if the added IISG is small, its addition should not significantly change the optimal strategy in the full game. Instead of restarting from scratch, we aim to preserve, as much as possible, what we have learned in Γ, without weakening CFR's long-term convergence guarantee.

The key idea to our approach is to act as if S had been in the abstraction the entire time, but that the actions in S_r leading to S were never played with positive probability. Since we never played the actions in S_r, we may have accumulated regret on those actions in excess of the bounds guaranteed by CFR. That would hurt the convergence rate. Later in this section we show that we can overcome this excess regret by discounting the prior iterations played (and thereby their regrets as well). The factor we have to discount by increases the higher the regret is, and the more we discount, the slower the algorithm converges because it loses some of the work it did. Therefore, our goal is to ensure that the accumulated regret is as low as possible. The algorithm we present can add actions for multiple players. In our description, we assume without loss of generality that the added actions belong to Player 1 (P_1).

We now discuss how to "fill in" what happened in IISG S during those T_0 iterations. Since the actions leading to S were never played, we have that for any history $h \notin S$ and any leaf node $z \in S$, $\pi^{\sigma^t}(h, z) = 0$ for all $t \in T_0$. By the definitions of counterfactual value (2) and regret (3) this means that regret for all actions in all information sets outside the greater subgame S^* remain *unchanged* after adding S. We therefore need only concern ourselves with S^*.

From (2) and (3) we see that $R^t(I, a) \propto \pi_{-i}^{\sigma^t}(h)$ for $h \in I$. Since S was never entered by P_1, for every history in S and every player other than P_1, $\pi_{-i}^{\sigma^t}(h) = 0$ and therefore $R^t(I, a) = 0$, regardless of what strategy they played. Normally, an information set's strategy is initialized to choose an action randomly until some regret is accumulated. This is obviously a poor strategy; had P_1's opponents played this way in S for all T_0 iterations, P_1 may have very high regret in S_r for not having entered S and taken advantage of this poor play. Fortunately, they need not have played randomly in S. In fact, they need not have played the same strategy on every iteration. We have complete flexibility in deciding what the other players played. Since our goal is to minimize overall regret, it makes sense to have them play strategies that would result in low regret for P_1 in S_r.

We construct an auxiliary game to determine regrets for one player and strategies for the others. Specifically, similar to the approach in [Burch *et al.*, 2014], we define a *recovery game* $S_i^{\sigma_\Gamma^{1..T}}$ for an IISG S for player i of a game Γ and sequence of strategy profiles $\sigma^{1..T}$. The game consists of an initial chance node leading to a history $h^* \in S_r$ with probability[1]

$$\frac{\sum_{t \in T} \pi_{-i}^{\sigma^t}(h^*)}{\sum_{h^* \in S_H} \sum_{t \in T} \pi_{-i}^{\sigma^t}(h^*)} \tag{5}$$

If this is undefined, then all information sets in S^* have zero regret and we are done. In each $h \in S_H$, P_i has the choice of either taking the weighted average counterfactual value of the information set $I = I(h)$, $\frac{\sum_{t \in T} v_i^{\sigma^t}(I)}{\sum_{t \in T} \pi_{-i}^{\sigma^t}(I)}$, or of taking the action leading into S, where play continues in S until a leaf node is reached. We play $T_S = \sum_{h^* \in S_r} \sum_{t \in T} \pi_{-i}^{\sigma^t}(h^*)$ iterations of CFR (or any other regret-minimization algorithm) in the recovery game.

We now define the *combined game* $\Gamma + S$ of a game Γ and an IISG S following play of a recovery game $S_i^{\sigma_\Gamma^{1..T}}$ as the union of Γ and S with regret and average strategy set as follows. For any action a belonging to I such that $I \to a \notin S$, $R_{\Gamma+S}(I, a) = R_\Gamma(I, a)$. Otherwise, if $P(I) \neq i$ then $R_{\Gamma+S}(I, a) = 0$. If $P(I) = i$ and $I \subseteq S$ then $R_{\Gamma+S}(I, a) = R_S(I, a)$. In the case that $I \to a \in S$ but $I \not\subseteq S$, then $R_{\Gamma+S}(I, a) = \sum_{t \in T_S} v_i^{\sigma_S^t}(I, a) - \sum_{t \in T} v_i^{\sigma_\Gamma^t}(I)$.

If $I \in \Gamma$ then $\bar{\sigma}_{\Gamma+S}^T(I) = \bar{\sigma}_\Gamma^T(I)$. Otherwise, if $P(I) = i$ then $\bar{\sigma}_{\Gamma+S}^T(I) = \vec{0}$, and if $P(I) \neq i$ then $\bar{\sigma}_{\Gamma+S}^T(I) = \bar{\sigma}_S^T(I)$.

[1] In two-player games without sampling of chance nodes, $\sum_{t \in T} \pi_{-1}^{\sigma^t}(h)$ is easily calculated as the product of $\sum_{t \in T} \pi_2^{\sigma^t}(I)$ for the last information set I of P_2 before h, and multiplying it by $\sigma_c(a'|h')$ for all chance nodes $h' \sqsubset h$ where $h' \to a' \sqsubseteq h$.

The theorem below proves that this is equivalent to having played a sequence of iterations in $\Gamma + S$ from the beginning. The power of this result is that if we play according to CFR separately in both the original game Γ and the recovery game $S_i^{\sigma_\Gamma^{1..T}}$, then the regret of every action in every information set of their union is bounded according to CFR (with the important exception of actions in S_r leading to S, which the algorithm handles separately as described after the theorem).

Theorem 1. *Assume T iterations were played in Γ and then T_S iterations were played in the recovery game $S_i^{\sigma_\Gamma^{1..T}}$ and these are used to initialize the combined game $\Gamma + S$. Now consider the uninitialized game Γ' identical to $\Gamma + S$. There exists a sequence of T' iterations (where $|T'| = |T_S||T|$) in Γ' such that, after weighing each iteration by $\frac{1}{|T_S|}$, for any action a in any information set $I \in \Gamma'$, $R_{\Gamma'}^{T'}(I, a) = R_{\Gamma+S}^{T}(I, a)$ and $\bar{\sigma}_{\Gamma'}^{T'}(I) = \bar{\sigma}_{\Gamma+S}^{T}(I)$.*

Proofs are presented in an extended version of this paper.

As mentioned earlier, if we played according to CFR in both the original abstraction and the recovery game, then we can ensure regret for every action in every information set in the expanded abstraction is under the bound for CFR, with the important exception of actions a in information sets $I \in S_r$ such that for $h \in I$, $h \rightarrow a \in S$. This excessive regret can hurt convergence in the entire game.

Fortunately, Brown and Sandholm [2014] proved that if an information set I exceeds a bound on regret, then one can discount all prior iterations in order to maintain CFR's guarantees on performance. However, their theorem requires all information sets to be scaled according to the highest-regret information set in the game. This is problematic in large games where a small rarely-reached information set may perform poorly and exceed its bound significantly. We improve upon this in the theorem below.

We will find it useful to define *weighted regret*

$$R_{T,T_2,i}^{w}(a) = w \sum_{t=1}^{T} r_{t,i}(a) + \sum_{t=1}^{T_2} r_{t,i}(a) \tag{6}$$

and *weighted average strategy*

$$p_{\sigma_{T,T_2,i}^{w}}(a) = \frac{w \sum_{t=1}^{T} p_{\sigma_{t,i}}(a) + \sum_{t=1}^{T_2} p_{\sigma_{t,i}}(a)}{wT + T_2} \tag{7}$$

Theorem 2. *Suppose T iterations were played in some game. Choose any weight w_i such that*

$$0 \leq w_i \leq \min\left\{1, \frac{|\mathcal{I}_i|\Delta_i^2|A_i|T}{\sum_{I \in \mathcal{I}_i} \sum_{a \in A(I)} \left(R_+^T(I, a)\right)^2}\right\} \tag{8}$$

If we weigh the T iterations by w_i, then after T' additional iterations of CFR, $R_{T+T',i}^{w_i} \leq |\mathcal{I}_i|\Delta_i\sqrt{|A_i|}\sqrt{w_i T + T'}$, where $T' > \max\{T, \frac{\max_I \sum_{a \in A(I)} R_+^T(I,a)^2}{\Delta^2 |A|}\}$.

Corollary 1. *In a two-player zero-sum game, if we weigh the T iterations by $w = \min_i\{w_i\}$, then after T' additional iterations, the players' weighted average strategies constitute a 2ϵ-equilibrium where*

$$\epsilon = \max_i \frac{|\mathcal{I}_i|\Delta_i\sqrt{|A_i|}}{\sqrt{wT + T'}}$$

While using the largest w that our theory allows may seem optimal according to the theory, better performance is achieved in practice by using a lower w. This is because CFR tends to converge faster than its theoretical bound. Say we add IISG S to an abstraction Γ using a recovery game to form $\Gamma + S$. Let S_i, Γ_i, and $(\Gamma + S)_i$ represent the information sets in each game belonging to player i. Then, based on experiments, we recommend using

$$w_i = \frac{\sum_{I \in \Gamma_i} \sum_a \left(R_+^T(I, a)\right)^2 + \sum_{I \in S_i} \sum_a \left(R_+^T(I, a)\right)^2}{\sum_{I \in (\Gamma+S)_i} \sum_a \left(R_+^T(I, a)\right)^2} \tag{9}$$

where the numerator uses the regret of $I \subseteq S_r$ before adding the IISG, and the denominator uses its regret after adding the IISG. This value of w_i also satisfies our theory.

3.1 Adding Actions with Regret Transfer

In certain games, it is possible to bypass the recovery game and add subtrees in $O(1)$. We accomplish this with an approach similar to that proposed by Brown and Sandholm [2014], who showed that regret can be transferred from one game to another in $O(1)$ in special cases.

Suppose we have an IISG S_1 in Γ and now wish to add a new IISG S_2 that has identical structure as S_1. Instead of playing according to the recovery game, we could (hypothetically) record the strategies played in S_1 on every iteration, and repeat those strategies in S_2. Of course, this would require huge amounts of memory, and provide no benefit over simply playing new strategies in S_2 through the recovery game. However, it turns out that this repetition can be done in $O(1)$ time in certain games.

Suppose all payoffs in S_1 are functions of a vector $\vec{\theta}_1$. For example, in the case of $\vec{\theta}_1$ being a scalar, one payoff might be $u_i(z_1) = \alpha_{i,z}\theta_1 + \beta_{i,z}$. Now suppose the payoffs in S_2 are identical to their corresponding payoffs in S_1, but are functions of $\vec{\theta}_2$ instead of $\vec{\theta}_1$. The corresponding payoff in S_2 would be $u_i(z_2) = \alpha_{i,z}\theta_2 + \beta_{i,z}$. Suppose we play T iterations and store regret in S_1^* as a function of $\vec{\theta}_1$. Then we can immediately "repeat" in S_2 the T iterations that were done in S_1 by simply copying over the regrets in S_1 that were stored as a function of $\vec{\theta}_1$, and replacing $\vec{\theta}_1$ with $\vec{\theta}_2$.

Unlike the method of Brown and Sandholm [2014], it is not necessary to store regret in the entire game as a function of $\vec{\theta}_1$, just the regret in S_1^*. This is because when we transfer to S_2, the "replaying" of iterations in S_2 has no effect on the rest of the game outside of S_2^*. Moreover, if $\vec{\theta}_1$ is entirely determined by one player, say P_1, then there is no need to store regret for P_2 as a function of $\vec{\theta}$, because P_2's regret will be set to 0 whenever we add an IISG for P_1.

This can be extremely useful. For example, suppose whenever P_1 takes an action that sets $\vec{\theta}_1$, that in any subsequent information set belonging to P_1, all reachable payoffs are multiplied by $\vec{\theta}_1$. Then subsequent regrets need not be stored as a function of $\vec{\theta}_1$, only the information sets that can choose $\vec{\theta}_1$. This is very useful in games like poker, where bets are viewed as multiplying the size of the pot and after P_1 bets, if P_2 does

not immediately fold, then every payoff is multiplied by the size of the bet. In that case, regret for an action need only be stored as a function of that action's bet size.

It is also not strictly necessary for the structure of the IISGs to be identical. If S_2 has additional actions that are not present in S_1, one could recursively add IISGs by first adding the portion of S_2 that is identical to S_1, and then adding the additional actions either with regret transfer internally in S_2, or with a recovery game. However, if S_2 has fewer actions than S_1, then applying regret transfer would imply that illegal actions were taken, which would invalidate the theoretical guarantees.

Typically, slightly less discounting is required if one uses the recovery game. Moreover, regret transfer requires extra memory to store regret as a function of $\vec{\theta}$. However, being able to add an IISG in $O(1)$ is extremely beneficial, particularly for large IISGs.

4 Exploitability Computation in Games with Large or Continuous Action Spaces

In two-player zero-sum games, we can quickly evaluate how close a strategy profile is to a Nash equilibrium by calculating the *exploitability* of the strategy for each player. If v_i^* is the value of a Nash equilibrium solution for player i, then exploitability of player i is $e_i(\sigma_i) = v_i^* - \min_{\sigma'_{-i} \in \Sigma_{-i}} u_i(\sigma_i, \sigma'_{-i})$. In order to calculate the exploitability in the full game of a player's abstraction strategy, it is necessary to define the player's strategy in situations that do not arise in the abstraction—because the opponent(s) (and perhaps also chance) may take actions that are not included in the abstraction. Typically, this is accomplished by mapping an action not in the abstraction to one that is. This is referred to as *action translation*, and empirical results have shown that the randomized pseudo-harmonic mapping [Ganzfried and Sandholm, 2013] performs best among techniques developed to date.

To calculate exploitability in a game, it is typically necessary to traverse the entire game. This is infeasible in large and infinite games. However, in situations where a player maps a range of actions to a single abstract action, it may be possible to express the exploitability of each action as a function whose maximum is easy to find. With that we can calculate exploitability in the original (unabstracted) game by traversing only the abstraction.

We now define one class of such games. Consider the case of an abstraction that maps a range of full-game actions $[L_I, U_I] \subset \mathbb{R}$ in I to a single abstract action a, and suppose an action $\theta \in [L_I, U_I]$ is taken. Suppose further that for every information set I' in the abstraction belonging to $P(I)$ and reachable from I following a, any payoff z that can be reached from I' has a payoff that is multiplied by θ. That is, for any history $h' \in I'$ and $z \in Z$ such that $\pi(h', z) > 0$, we have $u_{P(I)}(z) = \theta u'_{P(I)}(z)$. Since the abstraction maps all actions $\theta \in [L_I, U_I]$ to the same state, the abstraction will play identically regardless of which θ is chosen. With that in mind, since every reachable payoff is scaled identically, each choice of θ results in a strategically identical situation for $P(I)$. Thus, rather than choosing a specific θ, we can instead choose the entire range $[L_I, U_I]$. Our traversal will then return the entire expected payoff as a function of θ, and we can then choose the value that would maximize the function.

We use this approach in our full-game exploitability calculation, allowing us to calculate exploitability in a full game of infinite size.

5 Where and When to Add Actions?

In previous sections we covered how one can add actions to an abstraction during equilibrium finding. In this section, we discuss where in the game, and when, to add actions.

Each iteration of CFR takes $O(H)$ time. If useless IISGs are added, this will make each iteration take longer. We therefore need some method of determining when it is worthwhile to add an action to an abstraction. Generally our goal in regret-minimization algorithms is to keep average overall regret low. Thus, the decision of where and when to add an information set will depend on how best we can minimize overall regret. Regret in information sets where we play CFR is $O(\sqrt{T})$, while regret in information sets not played according to CFR is $O(T)$. So, intuitively, if an information set has low regret, we would do a better job of minimizing average regret by not including it in the abstraction and doing faster iterations. But as it accumulates regret in $O(T)$, eventually we could better minimize average regret by including it in the abstraction.

Following this intuition, we propose the following formula for determining when to add an action. Essentially, it determines when the derivative of summed average regret, taken with respect to the number of nodes traversed, would be more negative with the IISG added. It assumes that regrets for actions not in the abstraction grow at rate $\sim T$ while all other regrets grow at rate $\sim \sqrt{T}$.

Proposition 1. *Consider a game $\Gamma + S$ consisting of a main game Γ and IISG S. Assume a player i begins by playing CFR only in Γ, so that each iteration takes $O(|\Gamma|)$, but at any time may choose to also play CFR in S (after which each iteration takes $O(|\Gamma| + |S|)$). Assume that when playing CFR on an information set I, squared regret for an action a where $\forall h \in I$, $h \rightarrow a \in \Gamma$ grows by a fixed amount every iteration: $\left(R^{T+1}(I,a)\right)^2 = \left(R^T(I,a)\right)^2 + C_I$ for some constant C_I. Assume that for others actions $\left(R^T(I,a)\right)^2 = C_I T^2$. Then the optimal point to begin playing CFR in S is on the earliest iteration T where*

$$\frac{\sum_{I \in \mathcal{I}_{\Gamma,i}} R^T(I)}{|\Gamma|} < \frac{\sum_{I \in \mathcal{I}_{\Gamma,i}} R^T(I) + \sum_{I' \in \mathcal{I}_{S,i}} \left(2R^T(I') - \frac{R^T(I')}{T}\right)}{|\Gamma| + |S|}$$

This proposition relies on two important assumptions: 1) we know how fast regret will grow (and that it grows at the rate specified in the proposition), and 2) we can calculate regret for information sets outside the abstraction. Generally, it is not possible to precisely know the growth rate of regret. In our experiments, we use the rate of growth in regret up to the current iteration as an estimate of future growth. It is possible that better heuristics can be constructed depending on the domain. For example, using the rate of growth over

only the most recent iterations, or some weighted average of that form, may provide a more accurate measurement. In our experiments, we found that the speed of our algorithm can be enhanced by making the condition in Proposition 1 slightly stronger by increasing the left hand side by a small amount (1% was a good value in our experiments, as we will discuss).

We can estimate regret for information sets outside the abstraction. Suppose we want to calculate $\sum_{t \in T} v_i^{\sigma^t}(I, a)$ for some action a leading to an IISG not in our abstraction. The opponents must have some defined strategies following this action. We can calculate a best response against those strategies. Since we could have played that best response on each iteration, we can calculate an upper bound on regret for a by multiplying the counterfactual value from the best response by the number of iterations. This approach can be applied to any finite game, and can even be used to evaluate all actions in some infinite games, such as those defined in Section 4. In special cases, we can also use regret transfer to provide an instantaneous measure of regret using the approach described in Section 3.1.

In general, games exhibit *abstraction pathology*: a Nash equilibrium computed in a finer-grained abstraction can be further from the full-game Nash equilibruim than a Nash equilibrium computed in a coarser abstraction [Waugh *et al.*, 2009b]. Since the method described in this section examines regret in the *full* game when considering adding actions, it ensures eventual *convergence to a Nash equilibrium in the full game*! Any full-game action experiencing linear growth in regret would, by design, eventually be added to the abstraction. Thus, any "weak points" of the abstraction in the full game are quickly addressed by including them in the abstraction.

6 Removing Actions from an Abstraction

One potential problem with adding many actions to the abstraction is that some may later turn out to not be as important as we thought. In that case, we may want to remove actions from the abstraction in order to traverse the game faster. In general one cannot remove actions because they have positive probability in the average strategies in CFR.

However, there are situations where we can remove actions (from the abstraction and from the average strategy).

First, in some variants of CFR, such as CFR+, the final strategies have been shown empirically to converge [Tammelin, 2014], so one does not need to consider average strategies. In such algorithms we can simply choose to no longer traverse the IISG that we want to remove. Then, if our heuristic later suggests that the IISG should be played, we can add it back in and use the recovery game to "fill in" the iterations it skipped.

Second, CFR converges (in practice and in theory) even if we eliminate any finite number of past iterations. If we decide to discard some number of iterations, perhaps the first portion of a run, and an IISG is only reached in that portion, then we can remove the IISG from the abstraction.

Third, even in vanilla CFR, there are situations where we can effectively remove IISGs. If for every player i, the probability on iteration t of reaching history h, $\pi_i^{\sigma^t}(h)$, equals zero, then regret and average strategy will not be updated for any player. In that case, there is no need to traverse the descendants of h. If the path leading to a given IISG has, for each player, an action belonging to that player with sufficiently negative regret, then it may make sense to "archive" the IISG by removing it from memory and storing it on disk. If CFR updates the regrets on that path so that at least one player has positive probability of reaching the IISG, then we can bring the IISG back into memory. In the experiments we use only this third action removal method (and we actually do not use disk but RAM).

7 Experiments

We tested our algorithm on a game we coin *continuous Leduc Hold'em* (CLH), a modification of regular Leduc Hold'em [Southey *et al.*, 2005], a popular testbed for research due to its small size and strategic complexity. In CLH, there is a deck consisting of six cards. There are two suits, with each suit having three cards: Jack, Queen, and King. There are a total of two rounds. In the first round, each player places an ante of 1 chip in the pot and receives a single private card. A round of betting then takes place with a two-bet maximum, with Player 1 going first. A player may bet or raise any real amount between 1% of the pot and 100% of the pot. (There are no "chip stacks" in this game.) In the second round, a single public shared card is dealt, and another round of betting takes place. Again, Player 1 goes first, and there is a two-bet maximum following the same format. If one of the players has a pair with the public card, that player wins. Otherwise, the player with the higher card wins.

We created three fixed abstractions of CLH. All bet sizes were viewed as fractions of the pot. All abstractions contained a fold and call action. Abstraction *Branch-2* included a min bet and max bet at every information set. *Branch-3* additionally contained a bet size of $\frac{1}{3}$, selected according to the pseudo-harmonic mapping. *Branch-5* further contained $\frac{1}{7}$ and $\frac{3}{5}$, again selected by the pseudo-harmonic mapping.

We initialized the abstraction that was used in automated action addition to contain only the minimum and maximum possible bet at each information set (in addition to fold and call). We ran vanilla CFR on each abstraction. The automated abstractions considered adding actions every 5 iterations according to the heuristic presented in Section 5 using regret transfer to estimate regret. As mentioned in that section, the heuristic cannot exactly predict how regret will grow. We therefore also tested automated abstraction refinement with slightly stronger conditions for adding actions to the abstraction: Recovery-1.01 and Transfer both multiply the left term in the condition by 1.01. Such changes to the heuristic, of course, retain our theoretical guarantees. Recovery-1.0 and Recovery-1.01 use a recovery game to add IISGs as described in Section 3, while Transfer uses regret transfer as described in Section 3.1.

We calculated exploitability in the full continuous game assuming the randomized pseudo-harmonic action translation is used. Figure 1 shows that Recovery-1.01 outperformed all the fixed abstractions at every point in the run. Moreover, while the fixed abstractions leveled off in performance, the automated abstractions continued to improve throughout the run.

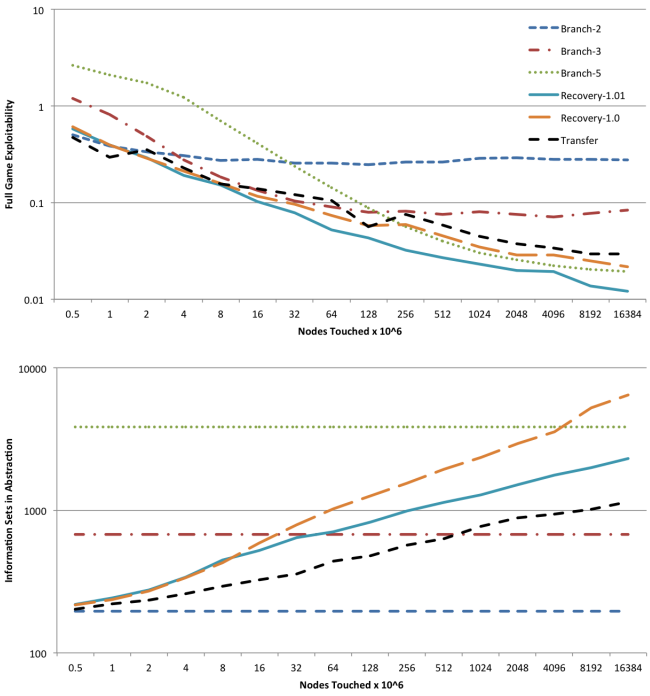

Figure 1: Top: Full game exploitability. Bottom: Abstraction size.

We also tested a threshold of 1.1, which performed comparably to 1.01, while a threshold of 2.0 performed worse.

Although regret transfer allows adding an IISG in $O(1)$ time, that method performed worse than using a recovery game. This is due to the regret from adding the IISG being higher, thereby requiring more discounting of prior iterations. The "bump" in the Transfer curve in Figure 1 is due to a particularly poor initialization of an IISG, which required heavy discounting. However, our heuristic tended to favor adding small IISGs near the bottom of the game tree. It is possible that in situations where larger IISGs are added, the benefit of adding IISGs in $O(1)$ would give regret transfer an advantage.

8 Conclusions

We introduced a method for adding actions to an abstraction simultaneously with equilibrium finding, while maintaining convergence guarantees. We additionally presented a method for determining strategic locations to add actions to the abstraction based on the progress of the equilibrium-finding algorithm, as well as a method for determining when to add them. In experiments, the automated abstraction algorithm outperformed all fixed abstractions at every snapshot, and does not level off in performance.

The algorithm is game independent, and is particularly useful in games with large action spaces. The results show that it can overcome the challenges posed by an extremely large branching factor in actions, or even an infinite one, in the search for a Nash equilibrium.

9 Acknowledgment

This work was supported by the NSF under grant IIS-1320620.

References

[Billings *et al.*, 2003] Darse Billings, Neil Burch, Aaron Davidson, Robert Holte, Jonathan Schaeffer, Terence Schauenberg, and Duane Szafron. Approximating game-theoretic optimal strategies for full-scale poker. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI)*, 2003.

[Brown and Sandholm, 2014] Noam Brown and Tuomas Sandholm. Regret transfer and parameter optimization. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2014.

[Brown *et al.*, 2015] Noam Brown, Sam Ganzfried, and Tuomas Sandholm. Hierarchical abstraction, distributed equilibrium computation, and post-processing, with application to a champion no-limit Texas Hold'em agent. In *International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, 2015.

[Burch *et al.*, 2014] Neil Burch, Michael Johanson, and Michael Bowling. Solving imperfect information games using decomposition. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2014.

[Ganzfried and Sandholm, 2013] Sam Ganzfried and Tuomas Sandholm. Action translation in extensive-form games with large action spaces: Axioms, paradoxes, and the pseudo-harmonic mapping. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 2013.

[Ganzfried and Sandholm, 2014] Sam Ganzfried and Tuomas Sandholm. Potential-aware imperfect-recall abstraction with earth mover's distance in imperfect-information games. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2014.

[Gibson, 2014] Richard Gibson. *Regret Minimization in Games and the Development of Champion Multiplayer Computer Poker-Playing Agents*. PhD thesis, University of Alberta, 2014.

[Gilpin and Sandholm, 2006] Andrew Gilpin and Tuomas Sandholm. A competitive Texas Hold'em poker player via automated abstraction and real-time equilibrium computation. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 1007–1013, 2006.

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

[Hawkin *et al.*, 2011] John Hawkin, Robert Holte, and Duane Szafron. Automated action abstraction of imperfect information extensive-form games. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2011.

[Hawkin *et al.*, 2012] John Hawkin, Robert Holte, and Duane Szafron. Using sliding windows to generate action abstractions in extensive-form games. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2012.

[Jackson, 2014] Eric Griffin Jackson. A time and space efficient algorithm for approximately solving large imperfect information games. In *AAAI Workshop on Computer Poker and Imperfect Information*, 2014.

[Johanson *et al.*, 2013] Michael Johanson, Neil Burch, Richard Valenzano, and Michael Bowling. Evaluating state-space abstractions in extensive-form games. In *International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, 2013.

[Johanson, 2013] Michael Johanson. Measuring the size of large no-limit poker games. Technical report, University of Alberta, 2013.

[Kroer and Sandholm, 2014a] Christian Kroer and Tuomas Sandholm. Extensive-form game abstraction with bounds. In *Proceedings of the ACM Conference on Economics and Computation (EC)*, 2014.

[Kroer and Sandholm, 2014b] Christian Kroer and Tuomas Sandholm. Extensive-form game imperfect-recall abstractions with bounds, 2014. arXiv.

[Lanctot *et al.*, 2012] Marc Lanctot, Richard Gibson, Neil Burch, Martin Zinkevich, and Michael Bowling. No-regret learning in extensive-form games with imperfect recall. In *International Conference on Machine Learning (ICML)*, 2012.

[Sandholm and Singh, 2012] Tuomas Sandholm and Satinder Singh. Lossy stochastic game abstraction with bounds. In *Proceedings of the ACM Conference on Electronic Commerce (EC)*, 2012.

[Sandholm, 2010] Tuomas Sandholm. The state of solving large incomplete-information games, and application to poker. *AI Magazine*, pages 13–32, Winter 2010. Special issue on Algorithmic Game Theory.

[Shi and Littman, 2002] Jiefu Shi and Michael Littman. Abstraction methods for game theoretic poker. In *CG '00: Revised Papers from the Second International Conference on Computers and Games*, pages 333–345, London, UK, 2002. Springer-Verlag.

[Southey *et al.*, 2005] Finnegan Southey, Michael Bowling, Bryce Larson, Carmelo Piccione, Neil Burch, Darse Billings, and Chris Rayner. Bayes' bluff: Opponent modelling in poker. In *Proceedings of the 21st Annual Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 550–558, July 2005.

[Tammelin, 2014] Oskari Tammelin. Solving large imperfect information games using CFR+. *arXiv preprint arXiv:1407.5042*, 2014.

[Waugh and Bagnell, 2015] Kevin Waugh and Drew Bagnell. A unified view of large-scale zero-sum equilibrium computation. In *Computer Poker and Imperfect Information Workshop at the AAAI Conference on Artificial Intelligence (AAAI)*, 2015.

[Waugh *et al.*, 2009a] Kevin Waugh, Nolan Bard, and Michael Bowling. Strategy grafting in extensive games. In *Proceedings of the Annual Conference on Neural Information Processing Systems (NIPS)*, 2009.

[Waugh *et al.*, 2009b] Kevin Waugh, David Schnizlein, Michael Bowling, and Duane Szafron. Abstraction pathologies in extensive games. In *International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, 2009.

[Waugh *et al.*, 2015] Kevin Waugh, Dustin Morrill, Drew Bagnell, and Michael Bowling. Solving games with functional regret estimation. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2015.

[Zinkevich *et al.*, 2007] Martin Zinkevich, Michael Bowling, Michael Johanson, and Carmelo Piccione. Regret minimization in games with incomplete information. In *Proceedings of the Annual Conference on Neural Information Processing Systems (NIPS)*, 2007.