Improving Performance in Imperfect-Information Games with Large State and Action Spaces by Solving Endgames*

Sam Ganzfried and Tuomas Sandholm
Computer Science Department
Carnegie Mellon University
{sganzfri, sandholm}@cs.cmu.edu

Abstract

Sequential games of perfect information can be solved by backward induction, where solutions to endgames are propagated up the game tree. However, this does not work in imperfect-information games because different endgames can contain states that belong to the same information set and cannot be treated independently. In fact, we show that this approach can fail even in a simple game with a unique equilibrium and a single endgame. Nonetheless, we show that endgame solving can have significant benefits in imperfect-information games with large state and action spaces: computation of exact (rather than approximate) equilibrium strategies, computation of relevant equilibrium refinements, significantly finer-grained action and information abstraction, new information abstraction algorithms that take into account the relevant distribution of players' types entering the endgame, being able to select the coarseness of the action abstraction dynamically, additional abstraction techniques for speeding up endgame solving, a solution to the "off-tree" problem, and using different degrees of probability thresholding in modeling versus playing. We discuss each of these topics in detail, and introduce techniques that enable one to conduct endgame solving in a scalable way even when the number of states and actions in the game is large. Our experiments on two-player no-limit Texas Hold'em poker show that our approach leads to significant performance improvements in practice.

1 Introduction

Sequential games of perfect information can be solved by a straightforward backward induction procedure in which solutions to endgames are propagated up the game tree. However, the same procedure does not work in general in games of imperfect information, and more sophisticated algorithms are needed. One algorithm for solving two-player zero-sum imperfect-information games is based on a linear program (LP) formulation (Koller, Megiddo, and von Stengel 1994). This formulation models each sequence of actions down the game tree as a variable and is called the *sequence-form* LP. It scales to games with around 10^8 states in the game tree. Many interesting games are significantly larger; for example, two-player limit Texas Hold'em has about 10^{17} game states, and a popular variant of two-player no-limit Texas Hold'em has about 10^{165} states (Johanson 2013). To address such large games, newer approximate equilibrium-finding algorithms have been developed that scale to at least 10^{12} states (Hoda et al. 2010; Zinkevich et al. 2007). These algorithms are iterative and guarantee convergence to equilibrium in the limit.

The leading approach for solving large games such as Texas Hold'em is to abstract the game down to a game with only 10^{12} states, then to compute an approximate equilibrium in the abstract game (Gilpin and Sandholm 2006; Billings et al. 2003; Sandholm 2010). In order to perform such a dramatic reduction in size, significant abstraction is often needed. *Information abstraction* involves reducing the number of game states by bundling signals (e.g., forcing a player to play the same way with two different hands). *Action abstraction* involves reducing the number of actions by discretizing large action spaces into a small number of actions.

1.1 Endgame Solving

A tempting technique to help mitigate the effects of abstraction and approximate-equilibrium finding is to solve relevant portions of the game that we actually reach during play separately online using a finer abstraction. We define an *endgame* E of game G as follows:[1]

Definition 1. *E is an* endgame *of game G if the following two properties hold:*

1. *If s' is a child of s and s is a state in E, then s' is also a state in E.*
2. *If s is in the same information set as s' and s is a state in E, then s' is also a state in E.*

For example, in poker we can consider endgames where several rounds of betting have taken place and several public cards have already been dealt (the rules of poker will be discussed in more detail in Section 2). In these endgames, we can assume both players have distributions of private

*This material is based upon work supported by the National Science Foundation under grants IIS-0964579 and CCF-1101668. We also acknowledge Intel Corporation and IBM for their machine gifts.

[1] An endgame is not the same as a *subgame*. In game theory, a subgame is a game rooted at a node (of the full game) that is alone in its information set.

information from states prior to the endgame that are induced from the base approximate-equilibrium strategy that we have precomputed in the coarse abstraction of the entire game. Given these distributions as inputs, we can then solve individual endgames in real time using much finer abstractions.

1.2 Theoretical Limitations of Endgame Solving

Unfortunately, this approach has some fundamental theoretical shortcomings. It turns out that even if we computed an exact equilibrium in the initial portion of the game prior to the endgame (which is an unrealistically optimistic assumption), and even if we are able to compute an exact equilibrium in the endgame, that the combined strategies for the initial game and endgame may fail to be an equilibrium in the full game. One obvious reason for this is that the game may contain many equilibria, and we might choose one for the initial game that does not match up correctly with the one for the endgame; or we may compute different equilibria in different endgames that do not balance appropriately. However, the following result shows that it is possible for this procedure to output a non-equilibrium strategy profile in the full game even if the full game has a unique equilibrium and a single endgame.

Proposition 1. *There exist games with a unique equilibrium and a single endgame for which endgame solving can produce a non-equilibrium strategy profile in the full game.*

Proof. Consider the Rock-Paper-Scissors game depicted in Figure 1. It has a single endgame—when it is player 2's turn to act. This game has a unique equilibrium—where each player plays each action with probability $\frac{1}{3}$. Now suppose we restrict player 1 to follow the equilibrium in the initial portion of the game. Any strategy for player 2 is an equilibrium in the endgame, because each one yields her expected payoff 0. In particular, suppose our equilibrium solver outputs the pure strategy Rock for her. This is clearly not an equilibrium of the full game. □

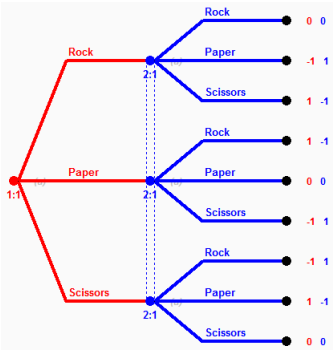

Figure 1: Sequential rock-paper-scissors with imperfect information.

1.3 Prior Work on Solving Endgames in Imperfect-Information Games

Despite this negative result, endgame solving has been incorporated into some agents for imperfect-information games. Limit Texas Hold'em agents PsOpti1 and PsOpti2 precomputed approximate equilibrium strategies for the final three betting rounds assuming a fixed strategy for the initial round (Billings et al. 2003). Several years later, the agent GS1 computed a real-time approximate equilibrium for the final two betting rounds using the LP formulation (Gilpin and Sandholm 2006); this was combined with a precomputed strategy for the first two rounds to produce a full strategy. However, this approach was soon replaced in GS2 which instead used a holistic approach that precomputed an approximate equilibrium for all four rounds in advance and did not perform any endgame solving (Gilpin, Sandholm, and Sørensen 2007). Endgame solving was again incorporated into the agent GS5 a few years later by solving the last betting round in real time, with a holistic approximate-equilibrium strategy precomputed for the prior rounds offline (Ganzfried and Sandholm 2010). This approach used a mixed integer programming formulation that took advantage of the qualitative structure of equilibrium strategies. While this approach was demonstrated to improve performance when applied to GS4, it actually led to a worse performance in GS5.

Applying finer-grained abstraction to different parts of the game tree repeatedly has been used as a way to try to improve equilibrium finding for the entire game offline, in limit Texas and Leduc Hold'em (Waugh, Bard, and Bowling 2009). This approach, known as *grafting*, was incorporated into the 2009 Hyperborean agent. A generalization of this, called *strategy stitching*, has been shown to be one viable way to tackle three-player limit Texas Hold'em by combining base strategies for the three-player game with solutions to two-player endgames (Gibson and Szafron 2011). Overall, progress on three-player poker is dramatically more nascent than in the two-player case, however.

Despite the attempts discussed above, by and large endgame solving has had very limited success in games of imperfect information. To the best of our knowledge, it is not used by any of the strongest agents for two-player limit Texas Hold'em. Furthermore, it has not been implemented by any competitive agents for games with large action spaces, such as no-limit Texas Hold'em (for any number of players).

1.4 Summary of the Rest of This Paper

In the rest of this paper, we describe several benefits of incorporating endgame solving for imperfect-information games with large state and action spaces. These benefits include significantly finer-grained state and action abstraction in the endgames, more sophisticated information abstraction algorithms that take into account type distributions induced by players' strategies before the endgames, exact computation of Nash equilibrium in the endgames, computation of relevant equilibrium refinements in the endgames, solving the 'off-tree problem' by correcting for ambiguities when the

opponent takes actions outside of our abstract action model for him before the endgames, using different degrees of probability thresholding in modeling and playing, and being able to select the coarseness of the action abstraction dynamically. We discuss each of these topics in detail, and describe techniques that enable one to conduct endgame solving in a scalable way in the large. Finally, experiments on two-player no-limit Texas Hold'em poker show that our techniques lead to significant performance improvements in practice.

2 No-Limit Texas Hold'em

Poker has received significant academic interest since the founding of the field of game theory (Nash 1951; von Neumann and Morgenstern 1947). This interest has been heightened in recent years due to the emergence of poker as a central AI challenge problem and the development of the Annual Computer Poker Competition. The most popular variant of poker among humans is no-limit Texas Hold'em. Two-player no-limit Texas Hold'em is played competitively by humans, and it is perhaps the game of most active research in the computer poker community currently. This game works as follows. Initially two players each have a *stack* of chips (worth $20,000 in the computer poker competition). One player, called the *small blind*, initially puts $50 worth of chips in the middle, while the other player, called the *big blind*, puts $100 worth of chips in the middle. The chips in the middle are known as the *pot*, and will go to the winner of the hand.

Next, there is an initial round of betting. The player whose turn it is to act can choose from three available options:

- *Fold:* Give up on the hand, surrendering the pot to the opponent.
- *Call:* Put in the minimum number of chips needed to match the number of chips put into the pot by the opponent. For example, if the opponent has put in $1000 and we have put in $400, a call would require putting in $600 more. A call of zero chips is also known as a *check*.
- *Bet:* Put in additional chips beyond what is needed to call. A bet can be of any size from 0 up to the number of chips a player has left in his stack (provided it is a multiple of the smallest chip denomination). A bet of all of one's remaining chips is called an *all-in* bet. If the opponent has just bet, then our additional bet is also called a *raise*. In some variants, the number of raises in a given round is limited, and players are forced to either fold or call at that point.

The initial round of betting ends if a player has folded, if there has been a bet and a call, or if both players have checked. If the round ends without a player folding, then three public cards are revealed face-up on the table (called the *flop*) and a second round of betting takes place. Then one more public card is dealt (called the *turn*) and a third round of betting, followed by a fifth public card (called the *river*) and a final round of betting. If a player ever folds, the other player wins all the chips in the pot. If the final betting round is completed without a player folding, then both players reveal their private cards, and the player with the best hand wins the pot (it is divided equally if there is a tie).

In particular, we will be interested in solving endgames after the final public card is dealt (but before the final round of betting). Thus, the endgame contains no more chance events, and only publicly observable actions of both players remain. As we show, these endgames are solvable in real time using the techniques we present.

3 Benefits of, and Techniques for, Endgame Solving

In imperfect-information games with large state and action spaces, endgame solving can have several benefits even though it may lose the equilibrium property. We describe several of these benefits in this section, and techniques that enable them in the large. The techniques are domain independent, but for concreteness we discuss them in the context of no-limit Texas Hold'em.

Recall that the overall approach is to first compute base strategies—one for each player—for the entire game offline (typically using abstraction-based approximation of game-theoretic equilibrium), and then to solve the endgame that is actually reached in more detail online. The inputs to our endgame solver are the current pot and stack sizes, the public cards on the table, and the hand distributions induced by the base strategies from the betting history using Bayes' rule.

3.1 Exact Computation of Nash Equilibrium in Endgames

The best algorithms for computing equilibria in large games of imperfect information scale to games with about 10^{12} states. However, these algorithms are iterative and guarantee convergence only in the limit; in practice they only produce approximations of equilibrium strategies. Sometimes the approximation error can be quite large. For example, one recent no-limit Texas Hold'em agent reported having an exploitability of 800 milli big blinds per hand (mbb/h) even within the abstract game (Ganzfried and Sandholm 2012). This is extremely large, since an agent that folds every hand would only have an exploitability of 750 mbb/h. An exact linear programming algorithm exists as well, based on the sequence-form formulation; however, it only scales to games with 10^8 states. While the LP algorithm is not applicable to large games like Texas Hold'em, we can use it to solve endgames that have up to 10^8 states exactly.

3.2 Computation of Equilibrium Refinements

The Nash equilibrium solution concept has some theoretical limitations, and several equilibrium refinement solution concepts have been proposed which rule out Nash equilibrium strategy profiles that are not rational in various senses. Common equilibrium refinements for extensive-form games of imperfect information include *undominated Nash equilibrium*, *perfect Bayesian equilibrium*, *sequential equilibrium*, *trembling hand perfect equilibrium*, and *proper equilibrium*. In general, these solution concepts guarantee that we behave sensibly against an opponent who does not follow his prescribed equilibrium strategy (e.g., he takes actions

that should be taken with probability zero in equilibrium). Specialized algorithms have been developed for computing many of these concepts (Miltersen and Sørensen 2008; 2010). However, those algorithms do not scale to large games. In Texas Hold'em, computing a reasonable approximation of a single Nash equilibrium already takes weeks or even months; so computing a refinement is clearly computationally infeasible. However, when solving endgames that are significantly smaller than the full game, it can be possible to compute a relevant equilibrium refinement.

In particular, we compute an undominated Nash equilibrium.

Definition 2. *A strategy s^* for player i* weakly dominates *strategy s' if for all pure strategies s_{-i} for the opponent,*

$$u_i(s^*, s_{-i}) \geq u_i(s', s_{-i}),$$

where the inequality is strict for at least one s_{-i}.

It seems intuitive that we would never want to play a dominated strategy s', since we could guarantee at least as good a payoff if we played some other strategy s^* instead. However, standard equilibrium-finding algorithms may find an equilibrium that is dominated. One example of this phenomenon is the equilibrium strategy profile computed by Gordon for one-card poker,[2] which used the LP formulation. In that equilibrium, player 1 always checks initially with a 5–8 and bets with positive probability with a 2–4 and 9–A. In response to a bet by player 1, player 2 always folds with a 2–4, always calls with a 9–A, and calls with probability strictly between 0 and 1 with 5–8. In particular, player 2 calls with a 5 with probability 0.251, with a 6 with probability 0.408, with a 7 with probability 0.583, and with an 8 with probability 0.759. Denote this strategy by s'. Now suppose player 2 instead called with a 5 with probability 0.01 and called with an 8 with probability 1, while keeping all other probabilities the same. Denote this new strategy by s^*. Clearly s^* weakly dominates s' (it performs strictly better if player 1 decides to bet with some probability with a 5, 6, 7, or 8). It is also clear that s^* also constitutes an equilibrium strategy for player 2.

It turns out that we can compute an undominated equilibrium with a relatively straightforward procedure (using the fact that any strategy that is a best response to a fully mixed strategy of the opponent is undominated (van Damme 1987)):

1. Compute the value of the game to player 1 (v_1) by solving the initial game.
2. Compute the best response of player 1 to a fully mixed strategy of player 2 subject to the constraint that an expected payoff of at least v_1 is attained in the worst case.
3. Compute the best response of player 2 to a fully mixed strategy of player 1 subject to the constraint that an expected payoff of at least $-v_1$ is attained.

Step 1 can be accomplished using the sequence-form LP formulation. Steps 2 and 3 can be solved using straightforward modifications of this LP. If our goal is just to play an undominated equilibrium strategy ourselves (which is often the case), we only need to perform Step 2 or Step 3 (depending on which player we are). Thus, overall this approach takes about twice as long as the standard approach for computing a Nash equilibrium. In our experiments, we used a uniform random strategy as the fully mixed strategy of the opponent.

[2]One-card poker is a simplified poker variant with a single round of betting, a single private card for each player from a 13-card deck, and a fixed bet size.

3.3 Finer-Grained Information and Action Abstraction

Perhaps the main benefit of endgame solving in games with large state and action spaces is that we can perform much finer abstraction in the endgame that is actually played than if we are forced to abstract the entire game at once in advance. In Texas Hold'em, there are many different information sets in the final betting rounds that must be compressed into a much smaller number of abstract information sets. When solving the endgame separately, we can solve it with no information abstraction at all quickly in real time for the particular sequence of publicly-observed actions and chance events. Rather than grouping all hands together into a small number of indistinguishable buckets, we can treat all distinct hands separately.

Significant action abstraction is also necessary to produce agents in no-limit Texas Hold'em. While agents are allowed to bet any integral number of chips (up to the number of chips remaining in their stack) at each opportunity, it is not computationally feasible to allow all such bet sizes. The standard approach is to discretize the betting spaces into a small discrete number of bet sizes for each history of publicly-observable actions. For example, the betting abstraction for a recent agent allows a fold, check/call (whichever is applicable), all-in, and at most 2 additional bet sizes at each game state (Ganzfried and Sandholm 2012). When solving endgames independently, significantly finer-grained betting abstractions can be used. For example, one of the betting abstractions we use (discussed in detail later) allows 8 different betting sizes for both players if no players have bet.

There is a clear tradeoff between information and action abstraction. Using more of one clearly means that we do not have to use as much of the other, and the optimal balance is application-specific. While full lossless information abstraction is possible in no-limit Texas Hold'em endgames, we decided to use some information abstraction so that we were able to use finer-grained action abstractions (as we will discuss in detail later).

3.4 New Algorithms for Strategy-Biased Information Abstraction

In this section we introduce a new technique for performing information abstraction and show that there is a natural way of applying it to endgame solving.

The standard approach for performing information abstraction in large imperfect-information games is to bucket information sets together for hands that perform similarly against a uniform distribution of the opponent's private information (Gilpin and Sandholm 2006; Gilpin, Sandholm,

and Sørensen 2007; Johanson et al. 2013).[3] However, the assumption that the opponent has a hand uniformly at random is extremely unrealistic in many situations; for example, if the opponent has called large bets throughout the hand, he is unlikely to hold a very weak hand. Ideally, a successful information abstraction algorithm would group hands together that perform similarly against the relevant distribution of hands the opponent actually has—not a naïve uniform random distribution.

Fortunately, we can accomplish such *strategy-biased information abstraction* in endgames.[4] In particular, our endgame abstraction algorithm takes as input the distributions of private information of both players (which, as described earlier in this paper, are induced using Bayes' rule by the approximate-equilibrium strategies precomputed for the entire game and the path of play so far in the game). Since there is no more public information to be revealed at the river (i.e., last betting round) in Texas Hold'em, we do not even need to store histograms for each hand; a single value will suffice to store the expected hand strength against the distribution of the opponent's hands. We compute the expected hand strength for each possible river hand for each player and perform clustering to compute an information abstraction of desired size.

3.5 Additional Information Abstraction Techniques for Speeding up Endgame Solving

To improve performance, we also applied another form of information abstraction by abstracting away the effects of *card removal*. When we arrive at the river endgame, the precomputed approximate equilibrium strategies induce a joint distribution over private cards of both players; however, the individual distributions are not independent. For example, if one player has AQ, then it is less likely the other player also has an ace or a queen. We ignore this card removal effect, and treat the distributions of players' hands as being independent; this assumption is fairly common in the poker literature (Ankenman and Chen 2006; Ganzfried and Sandholm 2010). This independence assumption allowed us to significantly reduce the runtime of our endgame solver. Without the assumption, we were forced to iterate over all possible values of the hole cards for both players. This *four-card rollout* was by far the bottleneck of the endgame solving, and using the independence assumption reduced runtime from about 45 seconds to about 3 seconds per endgame.

In order to make the computation fast, we use an additional technique for speeding up endgame solving. Our endgame solver first conducts information abstraction by ignoring card removal and *grouping hands together that have equal values at the given showdown* (e.g., 42 and 52 are equivalent on a board of AKJT8, since the best five-card hand is on the board). Next, it conducts strategy-biased information abstraction using the input distributions to obtain 20 card buckets for each player (in some cases, lossless abstractions were found with fewer buckets). To improve the quality of the information abstraction, our endgame solver does 10 repetitions of k-means++ (Arthur and Vassilvitskii 2007) and uses the clustering with lowest error out of the 10 results.

3.6 Solving the Off-Tree Problem

When we perform action abstraction, the opponent may take an action that falls outside of our action model for him. When this happens, an *action translation mapping* (aka reverse mapping) is necessary to interpret his action by mapping it to an action in our model (Ganzfried and Sandholm 2013; Schnizlein, Bowling, and Szafron 2009). However, this mapping may ignore relevant game state information. In poker, action translation works by mapping a bet of the opponent to a 'nearby' bet size in our abstraction; however, it does not account for the size of the pot or remaining stacks. For example, suppose remaining stacks are 17,500, the pot is 5,000, and our abstraction allows for bets of size 5,000 and 17,500. Now suppose the opponent bets 10,000, which we map to 5,000 (if we use a randomized translation mapping, we will do this with some probability). So we map his action to 5,000, and simply play as if he had bet 5,000. If we call his bet, we will think the pot has 15,000 and stacks are 12,500. However, in reality the pot has 25,000 and stacks are 7,500. These two situations are completely different and should be played very differently (for example, we should be more reluctant to bluff in the latter case because the opponent will be getting much better odds to call). This is known as the *off-tree problem*. Even if one is using a very sophisticated action translation algorithm, one will run into the off-tree problem.[5]

When performing endgame solving in real time, we can solve the off-tree problem completely. Regardless of the action translation used to interpret the opponent's actions prior to the endgame, we can take the stack and pot sizes (or any other relevant game state information) as inputs to the endgame solver. Our endgame solver in poker takes the current pot size, stack sizes, and prior distributions of the cards of both players as inputs. Therefore, even if we mapped the opponent's action to 5,000 in the above example, we correct the pot size to 25,000 before solving the endgame.

3.7 Dynamically Deciding the Granularity of Action Abstraction

Another advantage of endgame solving is that we can dynamically decide the granularity of the action abstraction

[3] Recent work has also considered an approach where the opponent's preflop hands are first grouped into several buckets, then hands for the later rounds are grouped together if they perform similarly against each of the preflop buckets. (Johanson et al. 2013).

[4] The idea of using computed strategies to guide abstraction (and iterating between abstraction and equilibrium finding) has already been proposed (Sandholm 2010), but there the idea was used very differently, namely to bucket together information sets where the equilibrium plays similarly and to separate other previously bucketed information sets.

[5] Some agents try to solve this problem by themselves taking an action that is designed specifically to get us back 'on-path'; however, this is not always desirable or even possible.

used for the endgame. We now present a way of accomplishing that. Again, the technique is domain independent, but we describe it in full detail in the context of no-limit Texas Hold'em so that our experiments are reproducible.

We precomputed a set of (three in our experiments) different action abstractions of varying granularities for the endgame. Each of the three allowed for fold, call/check, and all-in options at each situation. Our coarsest abstraction just allowed for a single additional action (similar to action abstractions currently used by most competitive agents). Our medium abstraction allowed for three bet sizes when no bets have been made yet on the river, 2 bet sizes after a single bet has occurred, and 1 bet size after a bet and raise have occurred. Our finest action abstraction had 8 bet sizes for the initial bet, four sizes for a raise, and two sizes for an additional raise. Thus, the medium and fine action abstractions are significantly finer grained than abstractions commonly used by no-limit Texas Hold'em agents.

To determine which action abstraction to use, we used the following procedure. First, we compute the number of betting sequences in each of the three abstractions for the relevant pot and stack sizes of the given endgame: let b_i denote this value for the i'th abstraction (where $i = 0$ is the coarsest one and $i = 2$ is the finest one). Let a denote the number of states for each player in our card abstraction in the endgame (in our experiments we used $a = 20$). Let T denote a threshold for the maximum number of action sequences allowable in the LP in order to compute a Nash equilibrium sufficiently quickly. If $a \cdot b_2 < T$, then we used the finest abstraction; otherwise if $a \cdot b_1 < T$ then we used the medium abstraction; otherwise we used the coarse abstraction.

3.8 Different Degrees of Thresholding for Playing and Modeling

Another important design decision is the degree of *thresholding* to use—both for our play, and for constructing the hand distributions of us and the opponent leading into the endgame. Thresholding is a technique used by most competitive agents where actions taken with low probability (below a specified threshold) are rounded down to zero, and remaining action probabilities are renormalized (Ganzfried, Sandholm, and Waugh 2012). This has been shown to successfully combat the problem of the equilibrium finder overfitting the strategies to the given abstraction, and to lead to significantly better performance in limit and no-limit Texas Hold'em. The extreme case of playing a deterministic strategy is called *purification*.

We used purification for our own play in all rounds, as it has previously been demonstrated to perform best. The obvious approach for constructing the prior hand distributions (one per player) that are inputs to the endgame solver would be to assume both players had been using purification as well. However, this proved to be problematic for several reasons. First, the opponent would occasionally take unusual action sequences that our purified agent would never take, and this would cause the prior distributions to be undefined (this could still occur even if no thresholding were used, but it is more likely to occur with a higher threshold). In addition, we observed that sometimes our hand distribution at the given endgame included just a single hand. In this case, our endgame equilibrium would assume that the opponent knew our exact hand, which is an extremely unrealistic and pessimistic assumption because in practice the opponent would not know our exact strategy. For these reasons, we decided to not use any thresholding when constructing the input distributions to the endgame solver—though we continued to use purification for ourselves when actually playing.

4 Example Demonstrating the Operation of our Endgame Solver

Before presenting our results, we first present an actual hand from our experiments that demonstrates how each step of a no-limit Texas Hold'em agent that uses our endgame solver works in practice. Recall that blinds are \$50 and \$100 and that both players start with \$20,000. Also recall that the strategies for our agent in the preflop, flop, and turn rounds are precomputed, and are not determined by our endgame-solving algorithm.

In this hand, we are in the big blind with KsAc. The opponent raises to \$264, we re-raise to \$1056, and he calls (there is now \$2112 in the pot). The flop is 2h8c6d, we check, the opponent bets \$915, and we call (there is now \$3942 in the pot). The turn is 7s, we check, he bets \$1482, and we call (there is now \$6906 in the pot). The river is As (pairing our ace), and it is our turn to act. The endgame-solving algorithm begins at this point.

The first step is to compute the distributions of hands both players could have, using Bayes' rule and the precomputed strategies for the first three betting rounds. For efficiency, we computed these distributions independently for both players ignoring card removal, as described in Section 3.5. This resulted in a probability distribution over 72 possible hand groupings for each player. Our particular hand KsAc, fell into group number 53 (where 0 is the worst possible hand and 71 is the best possible hand—T9 in this case). According to the distributions we computed, our hand group beats about 85% of the opponent's hands in this situation (by contrast, if the opponent had the same hand, he would beat about 71% of our hands). If we had used the standard uniform approach for information abstraction, we would only have a single hand strength for a given hand (i.e., its performance against a uniform random hand). But because of our new strategy-biased approach, described in Section 3.4, we are able to assign different strengths to the same hand which depend on the distribution of hands we believe the opponent has.

Next, we cluster each of these 72 hand groups into 20 clusters using k-means++ (separately for each player), as described in Section 3.5. The clustering occured almost instantaneously, producing an error[6] of 0.005 for player 1 (us) and 0.0007 for player 2 (the opponent). It placed our actual hand into cluster 17 (where 0 is the weakest and 19 is the strongest).

[6]The error is the sum of squares of differences between the strategy-biased expected hand strength of each hand and that of the mean of the cluster to which it is assigned.

Next, we determine which action abstraction to use, employing the technique described in Section 3.7. Recall that we have precomputed three different action abstractions of varying granularity for each pot size (this explicit dependence of our action abstraction on the actual pot size solves the off-tree problem, described in Section 3.6). For this particular pot size, the abstractions had 88, 28, and 16 betting sequences per player, respectively. Since $20 \cdot 88 = 1760$, the product of the number of abstract information states and the number of betting sequences is below our threshold of 7500 for the finest action abstraction; so the algorithm choses that one.

Next, the system computes an equilibrium in the endgame with the chosen information and action abstractions. We do this using CPLEX's dual simplex algorithm. For this particular hand, the solve took about 1.5 seconds. If we were computing an undominated equilibrium, we would need to solve a second LP, using the procedure described in Section 3.2.

To play our strategy, we look up the relevant action probabilities in the strategy vector we computed. In the actual hand, it is our turn to act first on the river. Our action abstraction has eight allowable actions at this point: check, 0.25 pot, 0.5 pot, 0.75 pot, pot, 1.5 pot, 2 pot, and all-in. Our computed strategy said to check with probability 0.52 and to bet 0.25 pot with probability 0.48. Since we are using purification for our own strategy (see Section 3.8), we round the probabilities to be 1 for check and 0 for 0.25 pot, and we choose to check. Our opponent then decided to go all-in, for a large bet of \$13,094 into a pot of \$6,906. In this situation, our strategy indicated to call with probability 1. We called and ended up winning the pot (the opponent had Qd4d).

5 Experiments

We ran experiments on two-player no-limit Texas Hold'em against several agents submitted to the 2012 Annual Computer Poker Competition (ACPC).

The time limit in the competition is 7 seconds per hand on average, and we determined that using $T = 7500$ kept us well within this limit. In fact, our agent took approximately 3 seconds per hand that went to the river, and only around 25% of hands make it to the river. Nearly all of the time spent by our algorithm was on solving the LPs, and the abstraction was performed essentially instantaneously. Our undominated equilibrium agent took approximately 6 seconds per hand that went to the river, which still kept us well within the time limit. As discussed in Section 3.4, if we had used the full joint distributions instead of assuming independence, the computation would have taken approximately 40 seconds longer per hand, and would not have been tractable. Using $T = 7500$, the procedure was almost always able to use the fine or medium betting abstraction, as described in Section 3.7.

For the base strategy for the entire game, we used our competition strategy from the 2012 ACPC, called Tartanian5 (which was computed using the standard paradigm of hand-crafted action abstraction and automated information abstraction, followed by approximate equilibrium computation in the abstract game). That is the base strategy that we used for the preflop, flop, and turn rounds. Once the river card was dealt, we computed an equilibrium in the corresponding endgame in real time using all the techniques described above. Thus, the only difference between our endgame-solving agent and Tartanian5 is how they play the final betting round. For the LP solver, we used CPLEX's dual simplex algorithm.

We experimented with two different versions of our endgame-solving agent: one that uses the standard LP approach to compute an equilibrium, and one that computes an undominated equilibrium.

We experimented against five of the top six agents from the 2012 ACPC (the remaining agent, neo.poker.lab, was not available for testing due to technical issues). For all opponents, we present results against both Tartanian5 and our new agents in order to gauge the performance improvement. The results are given in Table 1.

The results indicate that solving endgames led to improved performance against each of the opponents. In some cases this improvement was quite dramatic; against Tartanian5 and Sartre, our win rate improved by over 100 mbb/h. Furthermore, against each opponent, computing an undominated equilibrium for the endgame outperformed the approach of computing a potentially dominated one.

6 Conclusions and Future Work

We demonstrated that endgame solving can be successful in practice in imperfect-information games with large state and action spaces despite the fact that the strategy profile it computes is not guaranteed to be an equilibrium in the full game. We described several benefits and introduced techniques that enable one to conduct endgame solving in a scalable way in the large. Finally, we described a Texas Hold'em agent that successfully implements each of these features, and showed that these techniques led to a significant performance improvement against the top agents from the 2012 Annual Computer Poker Competition.

One direction for future research would be to study equilibrium refinements more extensively. For instance, perhaps it would be a better use of computational resources to compute a Nash equilibrium in a finer-grained abstraction instead of computing a refinement such as undominated Nash equilibrium. Or perhaps some other equilibrium refinements that we did not consider can be computed efficiently and lead to bigger improvements. In addition, while we showed that endgame solving can lead to a non-equilibrium full-game strategy profile in some games, it is possible that in interesting classes of games (perhaps even poker) it actually is guaranteed to find an equilibrium.

We would also like to consider implications of our approach for human poker play. When examining the endgame equilibrium strategies, we observed several behaviors that challenge conventional strategy. For example, we observed that many different bet sizes were used—ranging from small bets of $\frac{1}{4}$ pot all the way to large all-in bets for 30 times the pot. In contrast, humans generally utilize a small number of bet sizes, typically between $\frac{1}{2}$ pot and pot. We also observed that equilibrium strategies often involved significant amounts of randomization—sometimes players would bet 3 or even 4 different amounts with positive probability at a

	Opponent				
	Tartanian5	Hyperborean	Sartre	Hugh	Little.rock
Vanilla endgame solver	115 ± 35	-124 ± 44	214 ± 56	-48 ± 43	204 ± 58
Undominated endgame solver	120 ± 42	-105 ± 59	238 ± 76	-39 ± 63	273 ± 84
Tartanian5	—	-161 ± 36	56 ± 38	-102 ± 30	165 ± 63

Table 1: No-limit Texas Hold'em results against the strongest competitors from the 2012 computer poker competition. Our base agent is Tartanian5, and the other opponents are listed in the final four columns. The units are milli big blinds per hand, and the 95% confidence intervals are reported.

given information set (humans generally play deterministic strategies).

References

Ankenman, J., and Chen, B. 2006. *The Mathematics of Poker*. ConJelCo LLC.

Annual Computer Poker Competition. `http://www.computerpokercompetition.org/`.

Arthur, D., and Vassilvitskii, S. 2007. k-means++: The advantages of careful seeding. In *SODA*.

Billings, D.; Burch, N.; Davidson, A.; Holte, R.; Schaeffer, J.; Schauenberg, T.; and Szafron, D. 2003. Approximating game-theoretic optimal strategies for full-scale poker. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI)*.

Ganzfried, S., and Sandholm, T. 2010. Computing equilibria by incorporating qualitative models. In *International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*.

Ganzfried, S., and Sandholm, T. 2012. Tartanian5: A heads-up no-limit Texas Hold'em poker-playing program. In *Computer Poker Symposium at the National Conference on Artificial Intelligence (AAAI)*.

Ganzfried, S., and Sandholm, T. 2013. 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)*.

Ganzfried, S.; Sandholm, T.; and Waugh, K. 2012. Strategy purification and thresholding: Effective non-equilibrium approaches for playing large games. In *International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*.

Gibson, R., and Szafron, D. 2011. On strategy stitching in large extensive form multiplayer games. In *Proceedings of the Annual Conference on Neural Information Processing Systems (NIPS)*.

Gilpin, A., and Sandholm, T. 2006. A competitive Texas Hold'em poker player via automated abstraction and real-time equilibrium computation. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*.

Gilpin, A.; Sandholm, T.; and Sørensen, T. B. 2007. Potential-aware automated abstraction of sequential games, and holistic equilibrium analysis of Texas Hold'em poker. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*.

Gordon, G. One-card poker. `http://www.cs.cmu.edu/~ggordon/poker/`.

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

Johanson, M.; Burch, N.; Valenzano, R.; and Bowling, M. 2013. Evaluating state-space abstractions in extensive-form games. In *Autonomous Agents and Multi-Agent Systems*.

Johanson, M. 2013. Measuring the size of large no-limit poker games. Technical Report TR13-01, Department of Computing Science, University of Alberta.

Koller, D.; Megiddo, N.; and von Stengel, B. 1994. Fast algorithms for finding randomized strategies in game trees. In *Proceedings of the 26th ACM Symposium on Theory of Computing (STOC)*.

Miltersen, P. B., and Sørensen, T. B. 2008. Fast algorithms for finding proper strategies in game trees. In *Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*.

Miltersen, P. B., and Sørensen, T. B. 2010. Computing a quasi-perfect equilibrium of a two-player game. *Economic Theory* 42(1):175–192.

Nash, J. 1951. Non-cooperative games. *Annals of Mathematics* 54:289–295.

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

Schnizlein, D.; Bowling, M.; and Szafron, D. 2009. Probabilistic state translation in extensive games with large action sets. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI)*.

van Damme, E. 1987. *Stability and Perfection of Nash Equilibrium*. Springer-Verlag.

von Neumann, J., and Morgenstern, O. 1947. *Theory of Games and Economic Behavior*. Princeton University Press.

Waugh, K.; Bard, N.; and Bowling, M. 2009. Strategy grafting in extensive games. In *Proceedings of the Annual Conference on Neural Information Processing Systems (NIPS)*.

Zinkevich, M.; Bowling, M.; Johanson, M.; and Piccione, C. 2007. Regret minimization in games with incomplete information. In *Proceedings of the Annual Conference on Neural Information Processing Systems (NIPS)*.