Computing Strong Game-Theoretic Strategies and Exploiting Suboptimal Opponents in Large Games

Sam Ganzfried

Abstract

Designing successful agents in large multiagent strategic settings is a challenging problem for several reasons. First, many games of interest are far too large to be solved (for a relevant game-theoretic solution concept) by the best current algorithms. For example, no-limit Texas Hold'em has approximately 10^{165} states in its game tree, while the best algorithms for computing a Nash equilibrium only scale to games with around 10^{12} states. A second challenge is that it is not even clear that our goal should be computing a Nash equilibrium in the first place. In games with more than two players (or two-player games that are not zero sum), playing a Nash equilibrium has no performance guarantee. Furthermore, even in two-player zero-sum games, we can often obtain significantly higher payoffs by learning to exploit mistakes of a suboptimal opponent than by playing a Nash equilibrium.

The standard paradigm for addressing the first challenge is to first approximate the full game with a strategically similar, but significantly smaller game, and then to solve this smaller abstract game. All of this computation is done offline in advance, and the strategies are then looked up in a table for actual game play. We have developed new algorithms for improving each step of this paradigm. In particular, we propose new algorithms for computing equilibria in several classes of games and new techniques for performing game abstraction, as well as new approaches for addressing the problem of the strategies we compute being overfit to the abstraction and a new approach for addressing the problem of interpreting actions of the opponent that have been removed from the abstraction. We also propose approaches for extracting human-understandable knowledge from the computed strategy files.

In addition, we propose a new paradigm in which relevant portions of the game are solved in real time in much finer degrees of granularity than the abstract game which is solved offline. We have demonstrated that this new paradigm can lead to significantly stronger performance in no-limit Texas Hold'em. We also propose an approach that uses this real-time solver to create an endgame database which can be integrated with our offline equilibrium solver in order to dramatically improve the algorithm's performance, enabling us to solve games with significantly less abstraction for the initial betting rounds.

In the final portion of the proposal, we address the second challenge by proposing new algorithms for effectively learning to exploit unknown opponents in large games after only a small number of interactions. Furthermore, we propose new algorithms for exploiting opponents that are able to guarantee a good performance in the worst case even against strong dynamic opponents.

1 Background

A *game* is an abstract model of strategic interaction between multiple agents, or players. Formally, a *strategic-form game* G consists of a finite set of *players* $N = \{1, \ldots, n\}$, a finite set of *pure strategies* S_i for each player, and a *utility function* $u_i : \times S_i \to \mathbb{R}$ for each player. Here $\times S_i$ denotes the space of *pure strategy profiles*—vectors of pure strategies, one for each player. To play a game, each agent i simultaneously selects a pure strategy $s_i \in S_i$, and then receives a payoff of $u_i(s_1, \ldots, s_n)$. In general, players are allowed to randomize over their pure strategies, and need not play deterministically. Let Σ_i denote the space of probability distributions over S_i, which we call the *mixed strategy space* of player i. When each agent i

plays $\sigma_i \in \Sigma_i$, the expected payoff to player i is

$$u_i(\sigma_1, \ldots, \sigma_n) = \sum_{s_1 \in S_1} \cdots \sum_{s_n \in S_n} \left[u_i(s_1, \ldots, s_n) \prod_{j=1}^{n} \sigma_j(s_j) \right].$$

Note that we have overloaded the utility operator to be defined over $\Sigma = \times \Sigma_i$, the space of *mixed strategy profiles*. If the players are following the mixed strategy profile $\sigma \in \Sigma$, let σ_{-i} denote the vector of strategies taken by all players other than i, and let Σ_{-i} denote the space of mixed strategies for these players. The *support* of a mixed strategy σ_i is the set of pure strategies for player i played with nonzero probability under σ_i. Mixed strategy σ_i *weakly dominates* σ'_i if $u_i(\sigma_i, \sigma^*_{-i}) \geq u_i(\sigma'_i, \sigma^*_{-i})$ for all $\sigma^*_{-i} \in \Sigma_{-i}$, where the inequality is strict for at least one σ^*_{-i}.

If the other agents are playing strategy profile σ_{-i}, then a *best response* (aka *nemesis*) for player i is any strategy in $\arg\max_{\sigma'_i \in \Sigma_i} u_i(\sigma'_i, \sigma_{-i})$. A *Nash equilibrium* is a strategy profile σ such that σ_i is a best response to σ_{-i} for all i. Thus, in a Nash equilibrium, all players are simultaneously playing a best response to the strategy profile of the other agents, and no agent has an incentive to deviate to a different strategy given that the other agents follow the prescribed profile.

John Nash first introduced the Nash equilibrium in 1951, and in that paper he proved that a Nash equilibrium exists in every strategic-form game [47]. Subsequently, the Nash equilibrium has emerged as the central solution concept in the field of game theory. If all agents were perfectly rational, then we would intuitively expect them to follow a Nash equilibrium; if they instead followed a non-equilibrium strategy profile, then at least one agent could improve his performance by playing a different strategy, in which case it would not be rational for him to follow the prescribed strategy profile.

The Nash equilibrium solution concept is particularly compelling in a class of games known as two-player zero-sum games (aka *matrix games*). A two-player game is *zero sum* if $u_1(s) + u_2(s) = 0$ for all $s \in \times_i S_i$. These are fully non-cooperative, competitive games where one player's loss is exactly equal to the other player's gain. In this class of games, we have the following result, which is called the *minimax theorem*:

$$v^* = \max_{\sigma_1 \in \Sigma_1} \min_{\sigma_2 \in \Sigma_2} u_1(\sigma_1, \sigma_2) = \min_{\sigma_2 \in \Sigma_2} \max_{\sigma_1 \in \Sigma_1} u_1(\sigma_1, \sigma_2).$$

The minimax theorem was first published by John von Neumann [55] in 1928, several decades before Nash's existence theorem. This theorem states that there exists a unique value v^* such that player 1 can guarantee himself an expected payoff of at least v^* regardless of the strategy chosen by player 2, and similarly that player 2 can guarantee himself an expected payoff of at least $-v^*$ regardless of the strategy chosen by player 1. We refer to v^* as the *value* of the game. Sometimes we will write $v_1 = v^*$ as the value of the game to player 1, and $v_2 = -v^*$ as the value of the game to player 2. The *exploitability* of a strategy is the difference between the value of the game and worst-case performance against a nemesis.

In two-player zero-sum games, the Nash equilibrium strategies for player 1 are precisely those strategies that guarantee a worst-case expected payoff of at v^* (and similarly, the Nash equilibrium strategies for player 2 are precisely those strategies that guarantee a worst-case expected payoff of at least $-v^*$). For any non-equilibrium strategy for player 1, there exists some strategy for player 2 such that player 1's expected payoff is strictly less than v^*. Thus, Nash equilibrium strategies have a strictly better worst-case guarantee than all other strategies. If we assume players took turns having the role of player 1 and player 2, then a Nash equilibrium strategy would guarantee at least breaking even against any opponent, while for every non-equilibrium strategy, there exists some counter strategy against which it would lose.

An additional property of Nash equilibria in two-player zero-sum games is that they are *exchangeable*: if (σ_1, σ_2) and (σ'_1, σ'_2) are Nash equilibria, then (σ_1, σ'_2) and (σ'_1, σ_2) are also Nash equilibria. Thus, if player 1 follows his portion of the strategy profile from one Nash equilibrium, and player 2 follows his portion of the strategy profile from a different Nash equilibrium, the overall strategy profile played still constitutes a Nash equilibrium.

One final property of Nash equilibria in two-player zero-sum games is that they can be computed in polynomial time using a linear programming (LP) formulation [9]. This means that, at least in theory, an efficient procedure exists for computing a Nash equilibrium that will scale to large games. As we will see, this does not necessarily mean that we can compute a Nash equilibrium in a satisfactory amount of time for specific games we are interested in, which may be extremely large.

Unfortunately, none of these properties that make Nash equilibrium compelling in two-player zero-sum games hold in more general classes of games. In two-player general-sum games and games with more than two players, there can exist multiple equilibria, each yielding different payoffs to the players. If one player follows one equilibrium while other players follow a different equilibrium, the overall strategy profile is not guaranteed to be an equilibrium. And furthermore, if one player plays an equilibrium strategy, he could do arbitrarily poorly if the opponents do not follow their components of that same equilibrium. In addition, the problem of computing a Nash equilibrium in these game classes has recently been shown to be PPAD-complete, and it is widely conjectured that no efficient algorithms exist [8, 10]. So even if we wanted to play a Nash equilibrium in these games, we may not be able to compute one, even in relatively small games. In two-player general-sum and multiplayer games, the Nash equilibrium is a much less satisfactory solution concept than in two-player zero-sum games.

Even in two-player zero-sum games, the Nash equilibrium is not quite the end of the story. For one, algorithms may not scale to specific games we are interested in. Furthermore, we can often obtain a significantly higher payoff than the value of the game against suboptimal opponents who are not playing an equilibrium strategy. Against such opponents, it may be desirable to try to learn and exploit their mistakes rather than to simply follow a static equilibrium strategy. Of course, such *opponent exploitation* would be similarly beneficial in general-sum and multiplayer games as well.

Despite the theoretical limitations described above, we will follow traditional terminology and refer to the problem of computing an (approximate) Nash equilibrium of a game as *solving* the game. The first portion of the proposal will focus on new approaches for solving games, while the latter portion will address the problem of developing game-playing agents that potentially deviate from a Nash equilibrium strategy in order to exploit opponents' mistakes.

1.1 Extensive-form games

While the strategic form can be used to model simultaneous actions, another representation, called the *extensive form*, is generally preferred when modeling settings that have sequential moves. The extensive form can also model simultaneous actions, as well as chance events and imperfect information (i.e., situations where some information is available to only some of the agents and not to others). Extensive-form games consist primarily of a game tree; each non-terminal node has an associated player (possibly *chance*) that makes the decision at that node, and each terminal node has associated utilities for the players. Additionally, game states are partitioned into *information sets*, where the player whose turn it is to move cannot distinguish among the states in the same information set. Therefore, in any given information set, a player must choose actions with the same distribution at each state contained in the information set. If no player forgets information that he previously knew, we say that the game has *perfect recall*. A (behavioral) *strategy* for player i, $\sigma_i \in \Sigma_i$, is a function that assigns a probability distribution over all actions at each information set belonging to i.

In theory, every extensive-form game can be converted to an equivalent strategic-form game; however, there is an exponential blowup in the size of the game representation, and therefore such a conversion is undesirable. Instead, new algorithms have been developed that operate on the extensive form representation directly. It turns out that the complexity of computing equilibria in extensive-form games is similar to that of strategic-form games; a Nash equilibrium can be computed in polynomial time in two-player zero-sum games (with perfect recall) [41], while the problem is hard for two-player general-sum and multiplayer games.

For many years, the standard algorithm for computing an equilibrium in two-player zero-sum extensive-form games with perfect recall was a linear programming formulation [41]. This formulation works by modeling each sequence of actions for each player as a variable, and is often called the *sequence form LP* algorithm. It runs efficiently in practice, but only scales to games with around 10^8 states in their game tree, and runs into memory limitations for larger games.

Unfortunately, many interesting games have far more than 10^8 states in their game tree. To solve such games, newer algorithms have been developed that are able to scale to games with approximately 10^{12} states in their game tree. These algorithms are iterative and converge to a Nash equilibrium in the limit. While the LP algorithm is able to compute an exact equilibrium, in practice these iterative algorithms can only compute an approximate, or ϵ-, equilibrium. An *ϵ-equilibrium* is a strategy profile in which each player achieves a payoff of within ϵ of his best response.

Two main iterative algorithms have been used for solving these larger games. The first, called EGT, is based on a generalization of Nesterov's excessive gap technique [32]. Recently, a more scalable version has been developed that converges to an ϵ-equilibrium in $O(\ln(\frac{1}{\epsilon}))$ iterations [24]. The other algorithm, called *counterfactual regret minimization* (CFR), stores the cumulative regret of each action at each information set, contingent on the information set being reached [61]. At each iteration, each action is selected in proportion to its counterfactual regret. This algorithm is run against itself in self play, and the average strategy for each player is proven to converge to an equilibrium. CFR requires $O(\frac{1}{\epsilon^2})$ iterations to converge to an ϵ-equilibrium, though each individual iteration is much faster than an iteration of EGT. Several sampling schemes have been used that significantly improve the performance of CFR in practice in various classes of games [22, 23, 36, 43].

Both algorithms parallelize well, and have been shown to scale effectively in practice to very large games, such as Texas Hold'em. While EGT has a better asymptotic performance guarantee in terms of the number of iterations needed for convergence, each iteration of EGT takes much longer than each iteration of CFR. Overall, these algorithms have selective superiority, and it is not clear which will perform best on a given game. Unlike EGT, CFR can still be run on games that have imperfect recall, as well as two-player general-sum and multiplayer games, though there are no significant general theoretical guarantees in such settings [1, 21, 42, 60].

1.2 Other game representations

While the majority of this proposal will deal with strategic-form and extensive-form games, some parts will deal with other game representations. A *stochastic game* is a collection of games (often these are strategic-form games); the agents repeatedly play a game from this collection, and then transition probabilistically to a new game depending on the previous game played and the actions taken by all agents in that game. We will consider stochastic games where the individual *stage games* are themselves extensive-form imperfect-information games. Unlike extensive-form games, stochastic games have a potentially infinite duration. In *discounted stochastic games*, the expected payoff of a player is the weighted sum of payoffs from each iteration, where weights are multiplied by some constant factor λ at each time step; in *undiscounted stochastic games*, the expected payoff is the limit of the average iteration payoff. Prior algorithms are guaranteed to converge to an equilibrium in certain classes of stochastic games [6, 33, 40, 44, 58]. However, no known algorithms are guaranteed to converge to an equilibrium in three-player stochastic games (even in the zero-sum case). In fact, it is unknown whether a Nash equilibrium is even guaranteed to exist in undiscounted stochastic games with more than two players.

Continuous games generalize finite strategic-form games to the case of (uncountably) infinite strategy spaces. Many natural games have an uncountable number of actions; for example, games in which strategies correspond to an amount of time, money, or space. While Nash equilibria have been proven to exist in some classes of games, simple examples have also been constructed that do not contain an equilibrium. Algorithms have been developed for computing equilibria in certain subclasses of continuous games [53, 54, 57]; however, there are natural game classes for which neither the algorithms nor the existence results apply.

1.3 Poker

While all of the new algorithms and techniques we present in this proposal are domain-independent and apply to broad classes of games, we will primarily be evaluating them in the domain of Texas Hold'em poker. Poker has received significant academic interest since the founding of the field of game theory [47, 56]. 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 (ACPC) [3]. Two-player poker is a two-player zero-sum extensive-form games with perfect recall; therefore, the algorithms described in Section 1.1 will apply. We will be considering several variants of poker including no-limit Texas Hold'em, the most popular variant of poker among humans. 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. For further information about AI research in poker, we refer the reader to recent survey articles [49, 51].

Two-player no-limit Texas Hold'em 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 1 chip up to the number of chips a player has left in his stack, provided it exceeds some minimum value[1] and is a multiple of the smallest chip denomination (by contrast, in the limit variant, all bets must of a fixed size, which varies depending on the round). 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 five-card hand (out of his two private cards and the five public cards) wins the pot (it is divided equally if there is a tie).

[1]The minimum allowable bet size is the maximum of the big blind and the size of the previous bet in the current betting round (if one has occurred).

1.4 Standard paradigm for game solving

Two-player no-limit Texas Hold'em has about 10^{165} states in its game tree, while the limit variant has about has about 10^{17} game states [34]; so neither of these can be solved directly using EGT or CFR, which only scale to games with 10^{12} states. The traditional approach for solving games of this magnitude is depicted in Figure 1. First, the original game is approximated by a smaller *abstract game* that hopefully retains much of the strategic structure of the initial game. The first abstractions for two-player Texas Hold'em were manually generated [5], while current abstractions are computed automatically [25, 38]. For smaller games, such as Rhode Island Hold'em, abstraction can be performed losslessly, and the abstract game is actually isomorphic to the full game [27]. However, for larger games, such as Texas Hold'em, we must be willing to incur some loss in the quality of the modeling approximation due to abstraction.

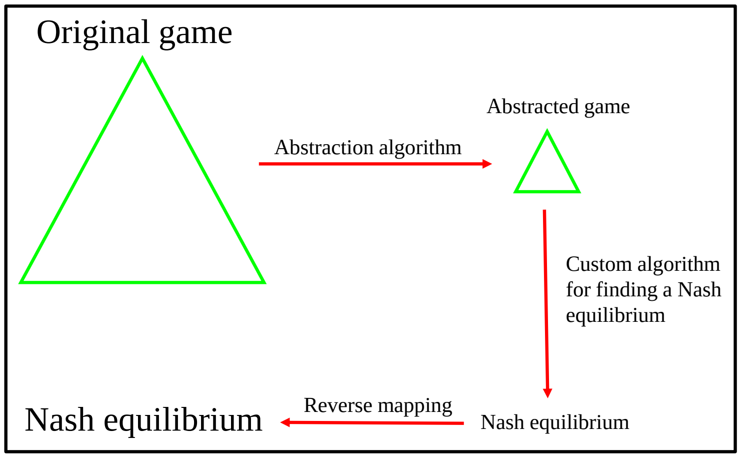


Figure 1: Traditional paradigm for solving large games.

In general, extensive-form games can have enormous strategy spaces for two primary reasons: the game tree has many information sets, or players have many actions available at each information set (e.g., when actions correspond to real numbers from some large set). There are two kinds of abstraction to deal with these two sources of complexity: *information abstraction* and *action abstraction*. In information abstraction, one groups information sets of a player together in order to reduce the total number of information sets. (Essentially this forces the player to play the game the same way in two different states of knowledge.) In action abstraction, one reduces the size of the action space. The typical approach for performing action abstraction is to discretize an action space into a smaller number of allowable actions; for example, instead of allowing agents to bid any integral amount between \$1 and \$1000, perhaps we limit the actions to only multiples of \$10 or \$100.

The second step in the traditional game-solving paradigm is to compute an ϵ-equilibrium in the smaller abstracted game, using a custom equilibrium-finding algorithm such as CFR or EGT.

The final step is to construct a strategy profile in the original game from the approximate equilibrium of the abstracted game by means of a *reverse mapping* procedure. When the action spaces of the original and abstracted games are identical, the final step is often straightforward, since the equilibrium of the abstracted game can be played directly in the full game. However, we will show that, even in this simplified setting, often significant performance improvements can be obtained by applying a nontrivial reverse mapping. In particular, we will introduce procedures, called *purification* and *thresholding*, that modify the action probabilities of the abstract equilibrium by placing more weight on the higher-probability actions [20]. These procedures are able to achieve robustness against overfitting strategies to a lossy abstraction and the failure of equilibrium-finding algorithms to fully converge within a given time limit.

When the action spaces of the original and abstracted games differ, an additional procedure is needed to interpret actions taken by the opponent that are not allowed in the abstract game model. Such a procedure is called an *action translation mapping*. The typical approach for performing action translation is to map the opponent's action to a nearby action that is in the abstraction (perhaps probabilistically), and then respond as if the opponent had taken this action.

While the first two steps have received significant attention over the last several years, the final step has received considerably little attention and is often overlooked. One of the main theses of this proposal is that the final step can be extremely important, and there are potentially significant benefits to a more rigorous and theoretically-principled study of reverse mapping. In fact, we will show that even with great abstraction and equilibrium-finding algorithms, the performance difference between using a naïve reverse mapping and using a more sophisticated one can be enormous.

2 New approaches for game solving within the standard paradigm

In this section, we propose new algorithms for improving each step of the standard paradigm. Section 2.1 will address the first step, Sections 2.2 and 2.3 will address the second step, and Sections 2.4 and 2.5 will address the third step. Finally, in Section 2.6, we propose approaches to obtain human-understandable knowledge from the computed strategy files.

2.1 New algorithms for computing imperfect-recall potential-aware information abstractions in large extensive-form games

The typical approach for performing information abstraction in large extensive-form games is to combine information sets together that are similar with respect to a relevant distance metric. Initially this was done manually [5], while current abstractions are computed automatically [25, 38]. For poker, the earliest abstraction algorithms grouped hands together at each round that had a similar *expected hand strength* against a uniform random hand for the opponent, assuming a random rollout of the future public cards and no additional betting in the later rounds [25, 26]. These approaches used clustering and integer programming to determine how many children to allocate to each information set at the next round.

One limitation of these approaches is that they do not capture the fact that hands can have different *potential*; for example, both 5s5h and QsJs will win approximately 60% of the time against a random hand for the opponent (assuming no more betting and a random rollout of community cards). However, 5s5h will usually be a mediocre hand that cannot be played very aggressively (since the board will often have several higher cards), while QsJs will often make a very strong hand (such as a straight or flush) or a very weak hand (Q high with no pair). We would like to group hands together that have similar *distributions* of hand strength—not just similar expected hand strength. This can be accomplished by creating a histogram for each hand, and grouping hands together if they have similar histograms. This was initially done doing a bottom-up pass of the tree, first computing an abstraction for the final round, then computing an abstraction for the third round taking into account distributions over buckets in the final round, etc. [28, 29]. However, more recent algorithms consider each round independently, and cluster hands together based on their histograms of hand strength assuming a uniform random hand for the opponent and the public cards, with no further betting [38].

Initially potential-aware abstractions were computed using the L2 distance metric [28, 29]. However, L2 is not ideal for comparing histograms, as the example in Figure 2 shows. The L2 distance between

Histogram 1 and Histogram 2 is

$$\sqrt{(1-0)^2+(0-1)^2+(0-0)^2+(0-0)^2+(0-0)^2}=\sqrt{2}.$$

However, the L2 distance between Histogram 1 and Histogram 3 equals $\sqrt{2}$ as well. The problem is that L2 distance does not take into account how far apart the entries of the histogram are. Recent work has shown the *earth mover's distance* (EMD) metric to be more useful in this setting [38]. Informally, EMD is the "minimum cost of turning one pile into the other; where the cost is assumed to be amount of dirt moved times the distance by which it is moved." The EMD between Histogram 1 and Histogram 2 is 1, while the EMD between Histogram 1 and Histogram 3 is 4.

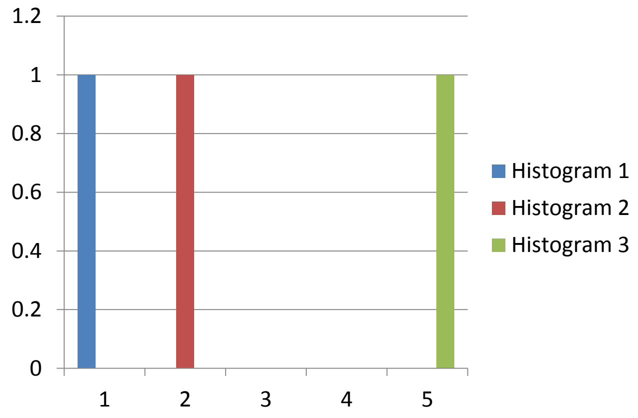


Figure 2: The L2 distance between Histogram 1 and Histogram 2 equals the L2 distance between Histogram 1 and Histogram 3, while the earth-mover's distances are different.

The initial algorithms created abstractions that had perfect recall; that is, they forced all players to remember exactly what private information they had in prior rounds [25, 26, 28, 29]. Recent algorithms have shown that using imperfect recall abstractions instead can lead to significant improvements in performance [38, 60]. As an informative example, consider the hands Ks3h and Ks2h on a board of KcQh9d. These two hands have top pair (kings), with a very low "kicker," that will very often be irrelevant (since only the best five card hand will win, out of the two private and five public cards). If we allow imperfect recall, then we can group these two hands together on the flop even if we were able to differentiate them on the preflop round. This allows much more flexibility in our choices of abstractions.

In summary, the state-of-the-art algorithms today use imperfect recall, earth-mover's distance, and take into account potential assuming uniform rollouts of all of the public cards and no further betting [38]. For the final betting round (aka the river), where there is no potential since there are no more public cards, the current approach is to first cluster the opponent's starting hands into some number k of clusters (typically 8), then make a vector of size k for the hand strength of each river hand against a uniform random hand in the given opponent cluster, and then to cluster these river hand-strength vectors using the L2 distance function [38].

While experiments have validated that imperfect recall performs better than perfect recall and EMD performs better than L2 (for the first three rounds, where we use histograms to take into account potential), we conjecture that the round-by-round bottom-up approach of earlier work [28, 29] is better than the current approach, since we care about how strong our hand will be on all future rounds, not just after the final round. The main reason this approach is not used is that the standard algorithm for computing the EMD between two histograms (where the x-axis corresponds to real numbers that have a well-defined ordering and distance metric) does not apply when we have histograms over buckets in the next round, which are not ordered.

We propose to address this by first presenting new algorithms for computing the EMD between two histograms over unordered buckets of the next round. One algorithm is an LP formulation, while the other is combinatorial (we expect the combinatorial one to perform better in practice). These algorithms assume no ordering over the buckets in the future round, but they assume a distance is given between each pair of buckets. We plan to use the distance (L2 for the river, and EMD for the earlier rounds) between the next-round bucket means as the distance function. We will then perform a bottom-up pass, starting with the turn, using the same clustering algorithm as before with the new algorithms for computing the earth-mover distances between points.

We also propose a new abstraction algorithm that will allow our implementation of counterfactual regret minimization to parallelize on the ccNUMA architecture, which uses shared memory. We will first cluster the public flop cards, and then compute abstractions separately on each public bucket, ensuring hands are only grouped together in a given round if their public cards are in the same public bucket. We will also ensure that buckets from one round only transition to buckets in the next round that have public cards in the same public bucket.

2.2 Computing equilibria in multiplayer stochastic imperfect-information games

We propose new algorithms for computing ϵ-equilibrium strategies in undiscounted multiplayer stochastic games, where the stage games are themselves extensive-form games of imperfect information [11, 12]. No prior algorithms are guaranteed to converge to an equilibrium in three-player stochastic games in this class of games, and it is still an open problem whether a Nash equilibrium is guaranteed to exist. We applied our algorithms to compute an ϵ-equilibrium of the three-player endgame of a poker tournament. While poker cash games, which are described in Section 1.3, are often modeled as extensive-form games, poker tournaments are often modeled as stochastic games because of their potentially infinite duration. In a poker tournament, players pay an entry fee and are given some number of tournament chips; a player is eliminated from the tournament when he runs out of chips, and prizes are awarded to the top finishers according to a prespecified payoff structure.

The most successful algorithm, called PI-FP, used a two-level iterative procedure, with a variant of policy iteration (an iterative algorithm for solving Markov decision processes) [48] in the inner loop and an extension of fictitious play (an iterative algorithm for solving and learning in games) [7] in the outer loop. Our main theoretical result is that if this algorithm converges, then the resulting strategy profile is a Nash equilibrium [12].

Proposition 1. *If the sequence of strategies $\{s^n\}$ determined by iterations of PI-FP converges, then the final strategy profile s^* is an equilibrium.*

We verified that our algorithms did in fact converge to an ϵ-equilibrium in a three-player poker tournament for very small ϵ (0.5% of the tournament entry fee). We were able to make several observations from the computed strategies, some of which challenged popular heuristics from the poker community [11]:

- A common method for evaluating the monetary value of tournament chip vectors, known as the Independent Chip Model, can sometimes lead to predictions that differ substantially from the expected payoffs in our approximate equilibrium strategy profile.

- There is no single fixed ranking of hands.

- Our strategies deviate significantly from a popular hand-ranking system, known as the Karlson-Sklansky system.
- Equilibrium strategies for playing a tournament and playing a cash game differ significantly when there are more than two agents (while they are similar for the case of two agents)
- Equilibrium strategies for three-player tournament endgames involve very little randomization.

2.3 Computing equilibria in continuous Bayesian games with small action spaces by incorporating qualitative models

We propose a new algorithm for computing equilibria in a class of continuous imperfect-information games where each agent has a private signal drawn from a compact subset X_i of $\mathbb{R}$ according to independent distributions F_i, and each agent has a finite action space C_i [13, 14]. The Nash existence theorem does not apply to this class of games, nor do existing algorithms for solving subclasses of continuous games. However, we show that natural games fall into this class, including simplified poker games that have previously been studied. While solving this general class of games is intractable, we show that it becomes tractable if we are given a qualitative model of the structure of an equilibrium. We have shown that if we are able to restrict strategies to conform to a qualitative model (e.g., by showing that the other strategies are dominated), then we can prove existence of an equilibrium. If we are given a candidate qualitative model, we have developed an algorithm that will compute an equilibrium consistent with the model if one exists, and will output that the problem is infeasible otherwise.

Proposition 2. *Given the distributions F_1, F_2 and a qualitative model, we have a complete mixed-integer linear feasibility program for finding an equilibrium.*

We have applied this approach to compute an equilibrium for a simplified model of the river betting round of two-player limit Texas Hold'em. (We assume that only one bet or raise is allowed per player in our model, and apply a heuristic to account for additional raises). We first constructed the three qualitative models shown in Figures 3- 5. The first one was constructed previously for the special cases when the F_i are the uniform distribution [2]. We constructed the final two by trial and error. Our experiments show that for every hand encountered in actual play in the poker competition, there was an equilibrium consistent with at least one of the three models. Furthermore, we showed that all three models were necessary to achieve this—no subset of two models sufficed. We showed that this approach led to an improved performance against our base equilibrium agent, and improved performance against 4 of the 5 entries from the 2008 ACPC.

We also have the following additional results:

- Algorithms for both the cases of discrete and continuous private signal distributions.
- An extension to the case when the private signals are dependent.
- An algorithm for computing an ϵ-equilibrium when there are more than two players.
- A new mixed integer programming formulation for computing an ϵ-equilibrium in strategic-form and extensive-form games with more than two players.
- An extension to the case when there are multiple qualitative models satisfying a technical condition.
- Experiments showing that solving a continuous approximation of a large finite game can outperform the standard approach of solving a smaller abstracted version of the game.

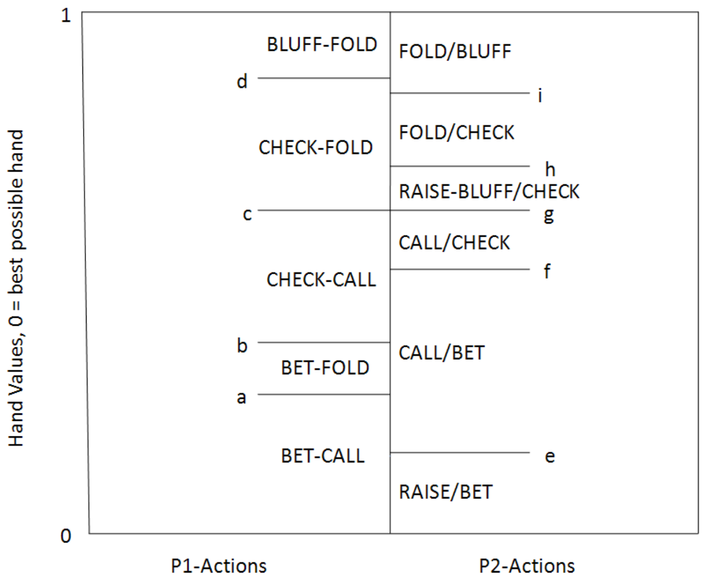


Figure 3: First qualitative model for limit Texas Hold'em.

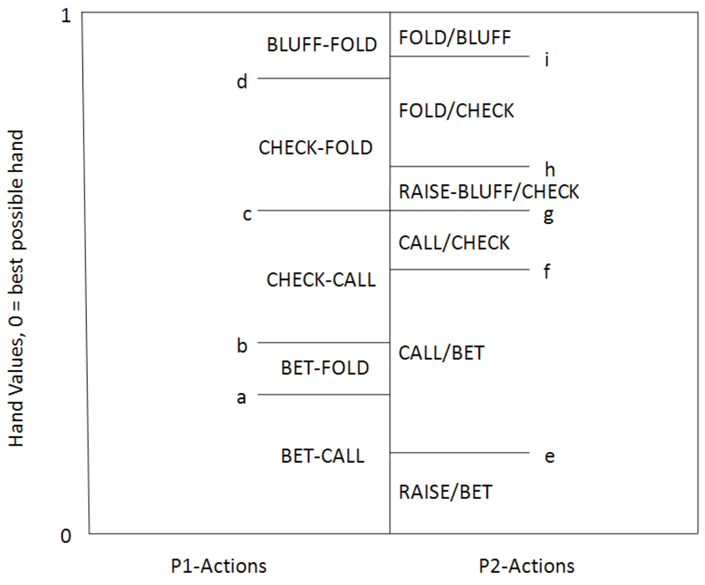


Figure 4: Second qualitative model for limit Texas Hold'em.

- Experiments demonstrating the convergence of our algorithm in a previously unsolved three-player game.

We think we have a new polynomial-time algorithm for finding an equilibrium given a qualitative model (which would be an exponential improvement over the MILP formulation). Remaining work will explore this approach further, theoretically and experimentally. We also plan to investigate the limit Texas Hold'em

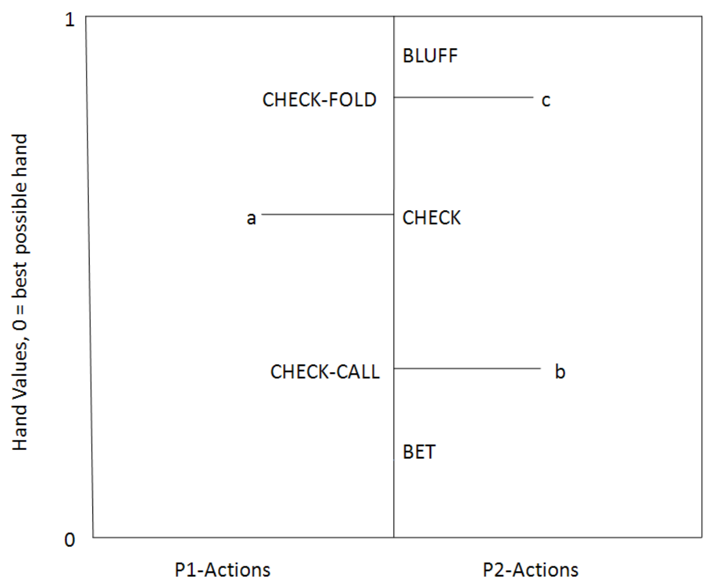


Figure 5: Third qualitative model for limit Texas Hold'em.

results further, and to characterize the situations in which each of the three models were used.

2.4 Strategy purification and thresholding

We next present new approaches for reverse mapping that are aimed at addressing the problems of overfitting the abstract equilibrium strategies to a lossy abstraction, and the failure of an equilibrium-finding algorithm to fully converge within a given time limit [20]. These have both been demonstrated to be very significant and real problems in large imperfect-information games. Figure 6 gives a graphical depiction of the overfitting phenomenon, showing that full-game exploitability can start to increase while abstract exploitability continues to decrease [35]. Additionally, we computed the exploitability of our entry in the 2012 two-player no-limit division of the ACPC, and determined that it was 800 mbb/hand even within its own abstraction [17]! (By contrast, folding every hand would have an exploitability of only 750 mbb/hand.) This indicates that the equilibrium-finding algorithm was very far from convergence.

We propose family of modifications to the standard approach that work by constructing non-equilibrium strategies in the abstract game, which are then played in the full game. If a mixed strategy σ_i plays a single pure strategy with highest probability, then the *purification* will play that strategy with probability 1. (If there is a tie between several pure strategies of the maximum probability played under σ_i, then the purification will randomize equally between all maximal such strategies). If σ_i is a behavioral strategy in an extensive-form game, we define the purification similarly; at each information set I, pur(σ_i) will play the purification of σ_i at I. We also consider a more relaxed approach, called *thresholding*, that only eliminates actions below a prescribed ϵ (the action probabilities are then renormalized so they sum to one).

We first show, in Proposition 3, that purified abstraction can perform arbitrarily better than abstraction alone against the full equilibrium strategy of the opponent. We can similarly show that purified abstraction can also do arbitrarily worse than unpurified abstraction, and that both procedures can do arbitrarily better or worse than thresholding (using any threshold cutoff). We can also show similar results using an arbitrary multiplicative (rather than additive) constant k. One motivation for using the full equilibrium strategy as the opponent's strategy is that many agents in the ACPC play static approximate equilibrium strategies.

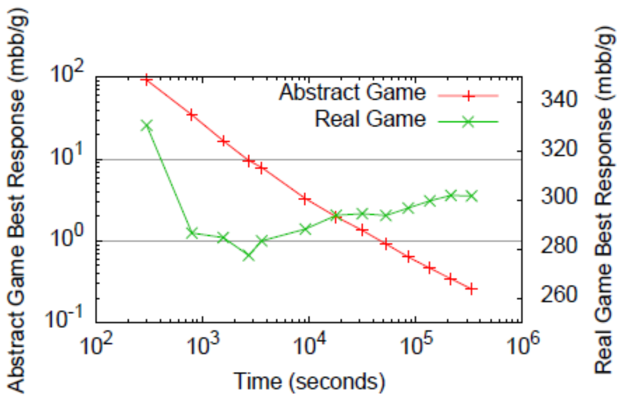


Figure 6: An example of overfitting using the standard game-solving paradigm.

Proposition 3. *For any equilibrium-finding algorithms A and A', and for any $k > 0$, there exists a game Λ and an abstraction Λ' of Λ, such that*

$$u_1(pur(\sigma_1'), \sigma_2) \geq u_1(\sigma_1', \sigma_2) + k,$$

where σ' is the equilibrium of Λ' computed by algorithm A', and σ is the equilibrium of Λ computed by A.

In random matrix games, we showed that purified abstraction outperforms the standard unpurified abstraction approach against the full equilibrium strategy of the opponent. We did this by simulating random 4×4 matrix games with payoffs drawn uniformly in $[-1, 1]$, and using abstractions that were the 3×3 games resulting from removing the final row and column. Results from these experiments are given in Table 1.

Observation 1. *Abstraction followed by purification outperforms abstraction alone against the full equilibrium strategy of the opponent in uniform random 4×4 matrix games using random 3×3 abstractions.*

$u_1(\text{pur}(\sigma_1^A), \sigma_2^F)$ (purified average payoff)	-0.050987 ± 0.00042
$u_1(\sigma_1^A, \sigma_2^F)$ (unpurified average payoff)	-0.054905 ± 0.00044
Number of games where purification led to improved performance	261569 (17.44%)
Number of games where purification led to worse performance	172164 (11.48%)
Number of games where purification led to no change in performance	1066267 (71.08%)

Table 1: Results for experiments on 1.5 million random 4×4 matrix games using random 3×3 abstractions. The $\pm$ given is the 95% confidence interval.

We present several general sets of conditions under which purified abstraction and unpurified abstraction lead to the same performance, which are given in Proposition 4. We further propose that the relative performances would not change if weakly dominated strategies were eliminated. We also observe a set of conditions on the support of the full game equilibrium for which purified abstraction leads to a higher, or equal, payoff to unpurified abstraction; these are given in Observation 2.

Proposition 4. *Let Λ be a two-player zero-sum game, and let Λ' be an abstraction of Λ. Let σ^F and σ^A be equilibria of Λ and Λ' respectively. Then*

$$u_1(\sigma_1^A, \sigma_2^F) = u_1(pur(\sigma_1^A), \sigma_2^F)$$

if either of the following conditions is met:

1. σ^A *is a pure strategy profile*
2. $support(\sigma_1^A) \subseteq support(\sigma_1^F)$

Observation 2. *In random* 4×4 *matrix games using* 3×3 *abstractions, pur*(σ_1^A) *performs better than* σ_1^A *using a 95% confidence interval for each support of* σ^F *except for supports satisfing one of the following conditions, in which case neither pur*(σ_1^A) *nor* σ_1^A *performs significantly better:*

1. σ^F *is the pure strategy profile in which each player plays his fourth pure strategy*
2. σ^F *is a mixed strategy profile in which player 1's support contains his fourth pure strategy, and player 2's support does not contain his fourth pure strategy.*

Experimental results indicate that purification and thresholding lead to significantly stronger play than the standard approach in several variants of poker. We submitted two programs to the no-limit Texas Hold'em division of the 2010 ACPC: Tartanian4-IRO (IRO) to the instant-runoff competition and Tartanian4-TBR (TBR) to the total bankroll competition. Both use the same abstraction and equilibrium-finding algorithms. They differ only in their reverse-mapping algorithms: IRO uses thresholding with a threshold of 0.15 while TBR uses purification. TBR performed better than IRO against every single opponent except for one, and beat IRO when they played head-to-head. We observed similar performance improvements of purification and aggressive thresholding in Leduc Hold'em, a simplified poker variant.

Furthermore, we observed that surprisingly more extreme thresholding does not necessarily produce more exploitable strategies. The exploitability of a purified version of our 2010 limit Texas Hold'em agent GS6 was significantly lower than the original version, while the minimum exploitability was attained using an intermediate threshold of 0.15. For Alberta's Hyperborean agent, exploitability increased monotonically with the threshold, as one might expect.

As remaining work, we hope to provide formal proofs of the results for matrix games given in Observations 1 and 2, as well as generalizations to games with arbitrary numbers of actions and abstraction sizes. We also plan to study the poker results further, and gain a better understanding of when and why purification and thresholding improve performance.

2.5 Action translation

When we perform action abstraction, we need a technique for determining how to respond when the opponent takes an action that has been removed from the model [18]. For example, we may have limited bids to multiples of \$100, but the opponent makes a bid of \$215. This is accomplished by an *action translation mapping*. Suppose the opponent bids $x \in [A, B]$, where A and B are the nearest action sizes in the abstraction. An action translation mapping corresponds to a function $f : [A, B] \rightarrow [0, 1]$, where $f(x)$ is probability we map x to A.

The obvious approach would be to simply map x to A if $x < \frac{A+B}{2}$, and otherwise map x to B. However, consider a natural poker situation where the pot is 1, $A = 1$, and $B = 100$.[2] For example, suppose our strategy calls a pot-sized bet of 1 with probability $\frac{1}{2}$ with a medium-strength hand (this probability is consistent with the analytical solution to many poker games [2]). If the opponent bets 1 with a very strong hand, his expected payoff will be $1 \cdot \frac{1}{2} + 2 \cdot \frac{1}{2} = 1.5$. However, if instead he bets 50, then his expected payoff will be $1 \cdot \frac{1}{2} + 51 \cdot \frac{1}{2} = 26$. So the opponent wins an additional \$24.50 by exploiting this mapping. In fact,

[2]The betting abstraction {fold, call, pot, all-in} is a common benchmark in no-limit poker [29–31, 52]: "previous expert knowledge [has] dictated that if only a single bet size [in addition to all-in] is used everywhere, it should be pot sized" [31].

this phenomenon was observed in the 2007 Annual Poker Competition when the agent Tartanian1 used this mapping and lost to an exploitative agent that did not even look at its private cards [29].

All of the prior action translation mappings that have been developed for poker are purely heuristic, and lack any theoretical justification. These include:

- Deterministic arithmetic: If $x < \frac{A+B}{2}$, then x is mapped to A; otherwise x is mapped to B (the mapping described above).
- Randomized arithmetic: $f(x) = \frac{B-x}{B-A}$.
- Deterministic geometric: If $\frac{A}{x} > \frac{x}{B}$ then x is mapped to A; otherwise x is mapped to B. Used by CMU's 2008 agent [29].
- Randomized geometric 1: $f(x) = \frac{A(B-x)}{A(B-x)+x(x-A)}$. Used by Alberta's 2011 agent [52].
- Randomized geometric 2: $f(x) = \frac{A(B+x)(B-x)}{(B-A)(x^2+AB)}$. Used by CMU's 2010 agent.

The randomized geometric mappings were the state of the art as of 2011, and even as of 2013 some strong agents continue to use one of them.

By contrast, we propose a new mapping that is theoretically motivated as a generalization of the analytical solution to a simplified poker game. Our mapping, called the *randomized pseudo-harmonic mapping*, is:

$$f(x) = \frac{(B-x)(1+A)}{(B-A)(1+x)}.$$

We show that our mapping produces much lower exploitability than the prior mappings in several variants of poker (the clairvoyance game, Kuhn poker, and Leduc Hold'em). For example, exploitabilities for the clairvoyance game [2] are given in Table 2. Furthermore, our new mapping performs competitively against no-limit Texas Hold'em agents submitted to the 2012 ACPC, in particular significantly outperforming the previously state-of-the-art randomized geometric approaches.

	Stack Size (n)						
	1	3	5	10	20	50	100
Det-Arith	0.01	0.24	0.49	1.12	2.38	6.12	12.37
Rand-Arith	0.00	0.02	0.09	0.36	0.96	2.82	5.94
Det-Geo	0.23	0.28	0.36	0.63	0.99	1.68	2.43
Rand-Geo-1	0.23	0.23	0.23	0.24	0.36	0.66	1.01
Rand-Geo-2	0.23	0.23	0.23	0.25	0.36	0.65	1.00
Det-psHar	0.15	0.19	0.33	0.47	0.59	0.67	0.71
Rand-psHar	0.00	0.00	0.00	0.00	0.00	0.00	0.00

Table 2: Exploitability of mappings for the clairvoyance game, using betting abstraction $\{$fold, check, pot, all-in$\}$.

We also propose a natural set of domain-independent desiderata that well-behaved translation mappings should satisfy.

1. **Boundary Constraints.** $f(A) = 1$ and $f(B) = 0$.
2. **Monotonicity.** f is non-increasing.
3. **Scale Invariance.** Scaling A, B, and x by some multiplicative factor $k > 0$ does not affect the mapping. (In poker for example, it is common to scale all bet sizes by the size of the big blind or the size of the pot).

4. **Action Robustness.** Small changes in x to not lead to large changes in f. (If f changes abruptly at some x^*, then the opponent could potentially significantly exploit us by betting slightly above or below x^*).

5. **Boundary Robustness.** Small changes in A or B do not lead to large changes in f (for all x). (If a tiny change in A (say from A_1 to A_2) caused $f_{A,B}(x)$ to change dramatically, then it would mean that f was incorrectly interpreting a bet of size x for either $A = A_1$ or $A = A_2$, and could be exploited if the boundary happened to be chosen poorly).

The following is a summary of our theoretical findings:

- The deterministic mappings violate action and boundary robustness.
- For $A = 0$, the geometric mappings are the constant function $f(x) = 0$, and violate the boundary condition $f(A) = 1$.
- The randomized geometric mappings violate boundary robustness. If we allow $A = 0$ they are discontinuous in A. Otherwise they are Lipschitz-discontinuous in A.
- Only the randomized arithmetic and randomized pseudo-harmonic mappings satisfy all of the desiderata. In particular, we show that they are Lipschitz continuous in A and B.

2.6 Extracting human-understandable knowledge from strategy files

Typically equilibrium-finding algorithms output strategies represented as a massive table, often in binary. While these can be easily implemented by a computer agent by a table lookup, it is difficult to extract any human-understandable knowledge from these strategy files (other than by simply observing the agent play for many hands). We propose to apply techniques from machine learning to extract knowledge from our strategy files that humans can actually understand. One approach would be to use algorithms from the Weka workbench, such as an algorithm for learning decision lists, to extract simple and interpretable rules from the files.

3 New paradigm for game solving

So far we have proposed new approaches for game solving that fit within the standard paradigm. In the standard paradigm, strategies are computed offline in advance, and the strategies are then looked up in a table for actual game play. We propose a new paradigm, depicted in Figure 7, in which relevant portions of the game are solved in real time in much finer degrees of granularity than the abstract game which is solved offline [19].

3.1 Improving performance in imperfect-information games with large state and large action spaces by solving endgames

We define an *endgame* E of game G as follows:[3]

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.

[3]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.

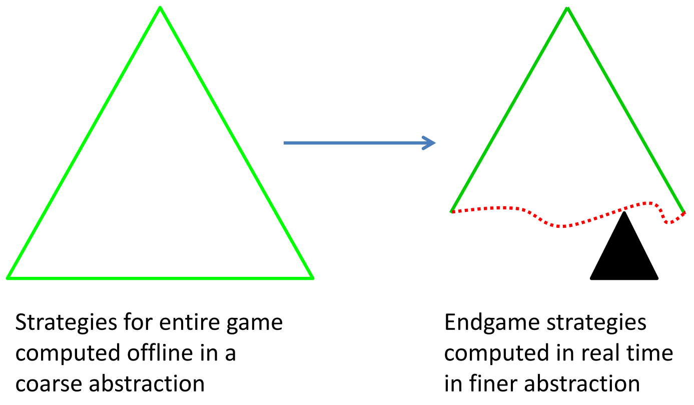


Figure 7: New paradigm for game solving by solving relevant portions of the game in real time.

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. In these endgames, we can assume both players have distributions of private 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.

A tempting technique to help mitigate the effects of abstraction and approximate-equilibrium finding is to solve the endgame that we actually reach during play separately online. 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, we show in Proposition 5 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 5. *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.*

Despite this negative result, we show that endgame solving has many theoretical benefits, and that it can lead to a significantly better performance in practice. These benefits include:

- **Computation of exact equilibria**. Iterative equilibrium-finding algorithms (such as CFR and EGT) guarantee convergence to equilibrium in the limit, but in practice the strategies they compute may be very far from equilibrium (for example, we observed this when our 2010 no-limit Texas Hold'em agent had a higher exploitability within its own abstraction than if it had folded every hand [17]). For endgames with up to 10^8 states, we can use the LP algorithm and obtain an exact equilibrium.

- **Computation of relevant equilibrium refinements**. Many equilibrium refinement solution concepts have been proposed, such as sequential, perfect, and proper equilibrium. Specialized algorithms have been developed for computing many of these concepts, but they do not scale to large games [45, 46]. When solving smaller endgames, it may be possible to compute a relevant equilibrium refinement. In particular, we can use an algorithm consisting of solving two LPs to compute an undominated equilibrium strategy.

- **Significantly finer-grained information and abstraction abstraction**.

- **New algorithms for strategy-biased information abstraction**. The standard approach for information abstraction is to bucket hands that perform similarly against a uniform distribution of hands of the opponent [38]. However, the assumption that the opponent has a hand uniformly at random is extremely unrealistic in many situations. When solving endgames separately, we can group hands together that perform similarly against the relevant distribution of hands the opponent actually has at the given situation.

- **Solving the "off-tree problem."** When the opponent takes an action that falls outside of our action model for him, we must apply an action translation mapping to interpret his action; however, this mapping may ignore relevant game state information. For example, consider the situation in no-limit Texas Hold'em where 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*. When performing endgame solving in real time, we can solve the off-tree problem completely.

- **Dynamically deciding the granularity of action abstraction**.

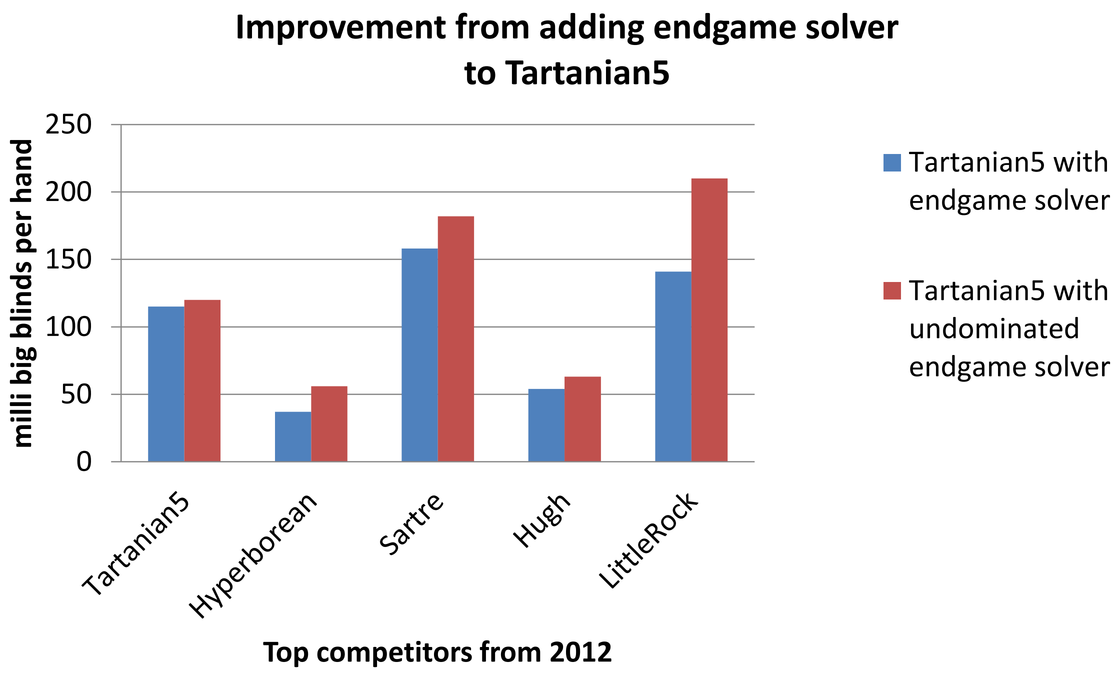
- **Different degrees of thresholding for playing and modeling**.

We observed that adding the endgame solver to our 2012 no-limit Texas Hold'em agent Tartanian5 led to an improved performance against the strongest competitors in the 2012 ACPC (Figure 3.1). Computing a non weakly-dominated equilibrium led to a further improvement against each opponent.

Proposed remaining work will investigate the tradeoff between computing an undominated equilibrium and computing a potentially dominated equilibrium in a larger abstraction (using the same time limit). We also plan to experiment with an algorithm for computing a quasi-perfect equilibrium [46], a refinement of undominated equilibrium. The algorithm just involves solving a single LP, but may be problematic due to numerical stability issues.

We also plan to further investigate the theoretical limitations of endgame solving. While we have shown it can produce strategies with extremely high exploitability in some games, it is possible that in interesting classes of games (perhaps a class that includes Texas Hold'em), it actually produces strategies with a very low exploitability. Proposition 6 shows that endgame solving can produce strategies with low exploitability in games where the opponent has a high degree of exploitative power within the endgames.

Proposition 6. *If every strategy that has exploitability of at least ϵ in the full game has exploitability of strictly more than δ within the endgame, then the strategy output by a solver that computes a δ-equilibrium in the endgame would constitute an ϵ-equilibrium of the full game.*



3.2 Extending the endgame solver to solve intermediate portions of the game

While we have primarily focused on solving endgames, the same approach could be used to solve intermediate portions of the game in real time as well. We propose to create a no-limit Texas Hold'em agent that solves each betting round (other than the first one) dynamically in real time, assuming that both agents are following the strategies that were precomputed offline. The input distributions to each round will be the distributions that were computed by the 'midgame' solver of the prior round. The main challenge will be to estimate the payoffs for the midgames. Possible approaches we will explore include precomputing a table, simulation, or using a machine learning algorithm.

Note that the endgame and midgame solvers takes arbitrary distribution of private signals as input. For our competition agent we used the approximate equilibrium strategy we precomputed as input for both agents. We also propose to integrate this approach with opponent exploitation and use a model as the input for the opponent's strategy. This is discussed further in Section 4.3. The endgame and midgame solvers could also be used in poker variants with more than two agents, as long as there are only two agents remaining in the relevant portion of the game (and we have input strategies to use).

3.3 An algorithm for efficiently computing equilibria in imperfect-information games with large state and large action spaces by incorporating an endgame database

We propose to create an endgame database consisting of the equilibria of the endgames from last year's computer poker competition. In addition, the database entries will store the pot size, and an encoding of the input distributions of private signals for both players. From the distributions, we can compute a vector of each player giving the equity of each private signal with respect to the opponent's input distribution of hands (i.e., the CDF with respect to the opponent's distribution). When a new endgame is encountered, its distance from each of the endgames in the database will be computed, and it will be mapped to the closest one. The distance metric will be the maximum over both players of the value of the earth mover distance between the CDF vectors. To look up the strategy once the closest endgame in the database is selected, we will look up the corresponding entry for the hand that is closest in equity to the given hand.

The endgame database will be computed offline. When an endgame is encountered in real time, we will just need to look up the closest endgame in the database rather than solve the new endgame in real time. Therefore, this will give a faster technique for approximating the solutions of endgames.

Furthermore, we plan to integrate this with our main CFR solver. Instead of sampling all the way down to the river during an iteration of MCCFR, we will sample down to the turn, then look up the equilibrium of the closest endgame in the database. This will significantly decrease the amount of memory needed, since we will not need to store strategies for the river betting round (which are larger than the strategies for the other rounds combined). Therefore, we will be able to use significantly larger abstractions for the first three rounds (possibly using no information abstraction at all).

4 Opponent exploitation

The previous portions of the proposal have focused on the problem of approximating Nash equilibrium strategies in large games. While equilibrium strategies guarantee a good performance in the worst case (at least in two-player zero-sum games), they potentially fail to take advantages of opponents' mistakes. We now turn to the problem of exploiting the mistakes of suboptimal opponents.

The natural approach for opponent exploitation is to try to learn a model of the opponent's strategy (based on prior game iterations and possibly additional historical data about the given opponent or other agents), then to maximally exploit this model. However, this approach has several drawbacks. First, it is incredibly difficult to learn an accurate model of the opponent's strategy quickly in large games. For example, no-limit Texas Hold'em has approximately 10^{165} states in its tree, while typical matches in the poker competition consist of just 3000 hands. So we only observe the opponent's actions at a minuscule portion of the information sets. The guarantees of no-regret learning algorithms are meaningless in such situations. This is further complicated by the fact that we may only see the opponent's private information after some hands and not others (e.g., in poker we only get to see the opponent's hand if no player folded in any betting round). For these reasons, opponent exploitation has been largely abandoned in recent years as a viable approach for strong agents in large imperfect-information games, in favor of the game-solving approach previously described.

An additional problem with this approach to opponent exploitation is that it can lead the exploiter to play a highly exploitable strategy himself. In general, a full best response will be a deterministic strategy that can be arbitrarily exploitable. This is problematic for several reasons. First, our model of the opponent will not be exact. If our model is wrong, we would like to not perform too poorly. Second, the opponent may not simply be playing a static strategy, and may try to deceive us by playing one way at the start of a match, then altering his strategy to exploit us once we start to exploit his initial strategy; this is known as the "get taught and exploited problem" [50]. A third reason is that if we play a maximal best response, it may be easier for the opponent to recognize his mistakes and fix his strategy (possibly exploiting us along the way).

Therefore, we would like to perform exploitation in a way that is robust to deviations of the opponent's strategy from our model. One approach that has been proposed is the *ϵ-safe best response* [37, 39]. In this approach, we assume the opponent plays according to our opponent model σ^* with some probability p, and that he can play arbitrarily with probability $1 - p$. Our strategy will be the solution to this new game. The parameter p is adjusted so that some desired level of exploitability ϵ for our own strategy is obtained.

This approach has been demonstrated to be successful in the setting where we have access to massive databases of historical play of our specific opponent with his private information labeled for each hand. It does the computation of the ϵ-safe best response offline in advance, using a standard equilibrium-finding algorithm such as CFR. Notice that a full best response can be computed much more efficiently than an ϵ-safe best response. A full best response can be computed by performing a matrix-vector product followed by a single pass up the game tree; by contrast, an ϵ-safe best response takes orders of magnitude longer (though both tasks can be done in polynomial time), requiring the computation of an equilibrium in the modified game. Therefore, this approach is intractable for real-time computation.

Furthermore, the assumptions of the approach are extremely strong. It is extremely rare to have a large amount of labelled data on the specific opponent at hand. Often we have unlabelled, or semi-labelled, data from the play of some pool of agents, which may or may not resemble the given opponent in any meaningful way. In many situations there may be no historical data available at all, and if we hope to exploit mistakes of an opponent, we must learn to do so in real time solely based on our observations of his play in prior game iterations.

A recent approach precomputes several exploitative strategies in advance using the procedure described above, then decides dynamically between then in real time using a no-regret algorithm [4]. However, this approach is also limited by the issues discussed above; specifically, it relies on having access to massive amounts of labelled data, and also relies on the fact that the given opponent will play similarly to the opponents represented in the dataset (and that we can determine which expert to play very quickly with partial observability of the opponent's private information).

We propose new approaches for opponent exploitation that do not rely on such strong (and often unrealistic) assumptions. They are able to effectively learn to exploit opponents in real time in large imperfect-information games after only a small number of interactions, without access to any historical data, and with partial or no observability of the opponent's private information in past game iterations. In addition, we show that in some games, it is actually possible to exploit suboptimal opponents significantly more than playing a static Nash equilibrium strategy while still guaranteeing zero exploitability in the worst case.

4.1 DBBR: A scalable, domain-independent opponent exploitation algorithm

Our first exploitation algorithm works by modeling the opponent using the strategy that is "closest" to the approximate equilibrium strategy we have computed in advance, subject to the constraint that it is consistent with our observations of the opponent's play thus far [15]. We consider several different measures of closeness, each of which can be computed efficiently in real time. Once we have constructed the opponent model, we compute a full best response to this model in real time, using a sufficiently coarse abstraction. Note that we can compute a full best response in real time in a reasonable abstraction in some games (such as limit Texas Hold'em), while computing an ϵ-safe best response would be infeasible. We repeat these steps of updating the opponent model and computing a best response as often as we can while staying under the given time limit (for the ACPC, the time limit is 7 seconds per hand on average throughout the match). We call the algorithm Deviation-Based Best Response (DBBR), since it builds the opponent model from observed deviations of the opponent's play from the approximate equilibrium strategy; pseudocode is given in Algorithm 1. While our exposition and experiments are for the two-player zero-sum case, we note that the algorithm applies just as well to non-zero-sum and multi-player games, since best response remains tractable for those game classes.

In more detail, we first partition all game states into *public history sets*, PH_i, where states in the same public history set correspond to the same history of publicly observed actions. In the first step of DBBR, an approximate equilibrium σ^* of the game is precomputed offline. Next, when the game begins, the frequencies of the opponent's actions at different public history sets are recorded. These are used to compute the probabilities with which he chooses each action at each public history set $n \in PH_{-i}$. We do not assume any

Algorithm 1 High-level overview of *DBBR*

```
Compute an approximate equilibrium of the game.
Maintain counters from observing opponent's play throughout the match.
for $n = 1$ to $|PH_{-i}|$ do
    Compute posterior action probabilities at $n$.
    Compute posterior bucket probabilities at $n$.
    Compute full model of opponent's strategy at $n$.
end for
return  Best response to the opponent model.
```

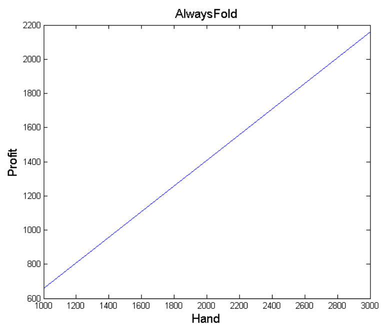
observations of the opponent's private information, and use a Dirichlet prior for initialization that assumes the opponent has played according to σ^* for some number of fictitious hands at each public history set (since we may have very few or no observations at some public history sets). Next, we compute the probability the opponent is in each bucket at n given our model of his play so far; these are computed in a single top-down pass using a BFS traversal order of the public history sets. We then compute a full model of the opponent's strategy by computing the closest strategy to σ^* that is consistent with our model of the opponent's action probabilities. We consider three different distance metrics; weighted L1 and L2 minimization (using CPLEX's LP and QP solvers respectively), and a new custom algorithm for minimizing the earth mover's distance. Finally, after we have constructed the full opponent model, we compute a best response.

We experimented in two-player limit Texas Hold'em against several naïve opponents, as well as several agents submitted to previous poker competitions. Matches consist of 3000 hands, and we play σ^* for the first 1000 hands before turning on the exploitation. Results are given in Figure 3. GS5 was our base approximate equilibrium strategy, which plays the role of σ^*. In nearly all the cases, each variant of our algorithm significantly outperformed GS5, while the earth-mover variant performed the best overall.

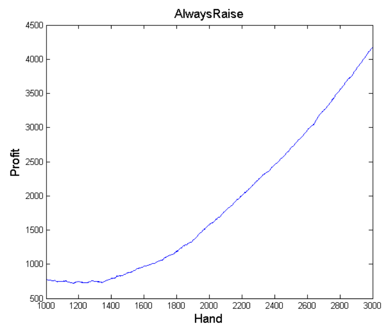
	Random	AlwaysFold	AlwaysCall	AlwaysRaise	GUS2	Dr. Sahbak	Tommybot
GS5	0.854 ± 0.008	0.646 ± 0.0009	0.582 ± 0.005	0.791 ± 0.009	0.636 ± 0.004	0.665 ± 0.027	0.552 ± 0.008
DBBR-EM	1.769 ± 0.025	0.719 ± 0.002	0.930 ± 0.014	1.391 ± 0.034	0.807 ± 0.011	1.156 ± 0.043	1.054 ± 0.044
DBBR-L_1	2.164 ± 0.036	0.717 ± 0.002	0.935 ± 0.017	0.878 ± 0.032	0.609 ± 0.054	1.153 ± 0.074	
DBBR-L_2	2.287 ± 0.046	0.716 ± 0.002	0.931 ± 0.026	1.143 ± 0.084	0.721 ± 0.050	1.027 ± 0.072	

Table 3: Win rate in small bets/hand of the bot listed in the row. The $\pm$ given is the standard error (standard deviation divided by the square root of the number of hands). The final opponent was not available for testing against each agent due to technical issues on the competition testing server.

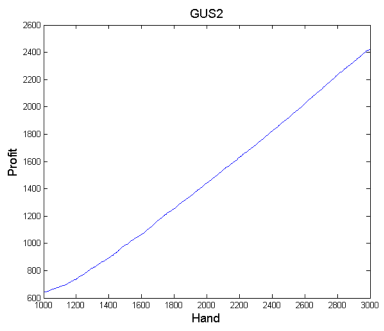
We expected that our win rate would increase steadily starting at hand 1001—when we begin the exploitation. This happened in most of the matches. For example, Figure 8(a) shows that DBBR's profits against AlwaysFold increase linearly over time, and Figure 8(d) shows that DBBR's win rate increases in a concave fashion. However, we observed a different behavior in the matches against AlwaysRaise and GUS2. In both of these matches, the win rate decreases significantly for the first several hundred hands before it starts to increase, as shown in Figure 8. This happens because the approximate-equilibrium strategy plays some action sequences with very low probability, leading it to not explore the opponent's full strategy space in the first 1000 hands. This will lead to a significant disparity between the prior and actual strategies of the opponent at hand 1001 if the opponent's strategy differs significantly from the approximate equilibrium in those unexplored regions. This in turn may cause DBBR to think it can immediately exploit the opponent in certain ways, which turn out to be unsuccessful; but eventually as DBBR explores these sequences further and gathers more observations, it figures out successful exploitations.



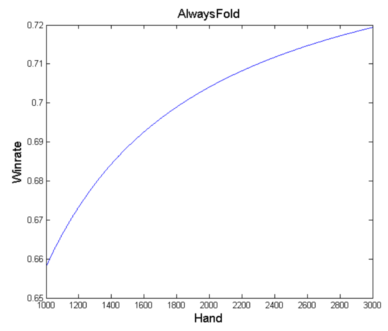
(a)



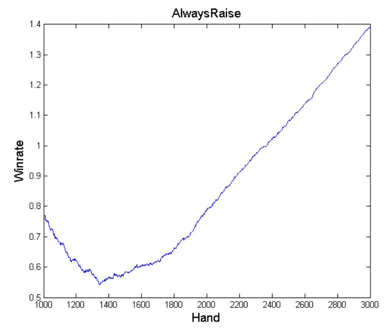
(b)



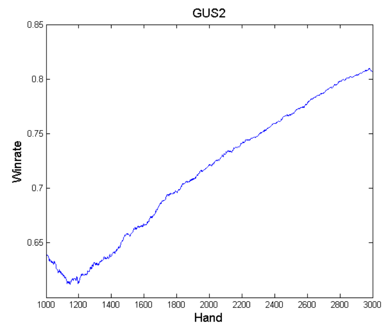
(c)



(d)



(e)



(f)

Figure 8: Profits and win rates over time of DBBR-EM against several opponents. Results against AlwaysFold are shown in Figures 8(a) and 8(d), results against AlwaysRaise are shown in Figures 8(b) and 8(e), and results against GUS2 are shown in Figures 8(c) and 8(f). The top three graphs show profit over time, and the bottom three show win rates over time.

4.2 Safe opponent exploitation

One limitation of DBBR is that it plays a full best response, and could perform very poorly against a strong dynamic opponent. We propose a new methodology for opponent exploitation that addresses this limitation. We show that in some games, it is possible to exploit the opponent for significantly greater profits than a static equilibrium strategy while still guaranteeing the value of the game in expectation in the worst case [16]. While our analysis will be for two-player zero-sum games, the analysis and approaches will apply to non-zero-sum and multiplayer games as well if we compare our payoff to the maximin value (aka security level), rather than the minimax game value (which is not defined in such games).

We first observe that in some games, such as rock-paper-scissors, it is not possible to deviate from equilibrium while still guaranteeing the value of the game in the worst case. However, if we give the opponent an additional pure strategy T that is strictly dominated (Figure 9), then it is possible to deviate from the stage-game equilibrium and still guarantee the value; for example, we could play R if the opponent played T in the prior iteration, and play the equilibrium otherwise. Such safe exploitation is possible because of the presence of the 'gift' strategy T.

	R	P	S	T
R	0	-1	1	4
P	1	0	-1	3
S	-1	1	0	3

Figure 9: Payoff matrix of RPST.

Recent work has conjectured the following:

Conjecture 1. *[59] An equilibrium strategy makes an opponent indifferent to all non-[weakly]-iteratively-dominated strategies. That is, to tie an equilibrium strategy in expectation, all one must do is play a non-[weakly]-iteratively-dominated strategy.*

This conjecture would seem to imply that gifts correspond to strategies that put weight on pure strategies that are weakly iteratively dominated. However, we present a counterexample to this conjecture, from which we have Proposition 7. In Proposition 8, we invalidate another candidate definition of gift strategies.

Proposition 7. *It is possible for a strategy that survives iterated weak dominance to obtain expected payoff worse than the value of the game against an equilibrium strategy.*

Proposition 8. *It is possible for a strategy that is not in support of an equilibrium to obtain the value of the game against an equilibrium strategy.*

Having ruled out several candidate definitions of gift strategies, we now present our new definition, which we relate formally to safe exploitation in Proposition 9.

Definition 2. *A strategy σ_{-i} is a* gift strategy *if there exists an equilibrium strategy σ_i^* for the other player such that σ_{-i} is not a best response to σ_i^*.*

Proposition 9. *Assuming we are not in a trivial game in which all of player i's strategies are minimax strategies, then non-stage-game-equilibrium safe strategies exist if and only if there exists at least one gift strategy for the opponent.*

One natural exploitation algorithm is 'Risk What You've Won' algorithm (RWYW); at each iteration, maximally exploit the opponent subject to risking only the amount of profit won so far. More specifically, at each iteration t, RWYW plays an ϵ-safe best response to a model of the opponent's strategy (according to some opponent modeling algorithm M), where ϵ is our current cumulative payoff minus $(t-1)v^*$. Pseudocode is given in Algorithm 2. We show in Proposition 10 that RWYW is not safe; intuitively, this is because it does not adequately differentiate between whether profits were due to skill (i.e., from gifts) or to luck.

Algorithm 2 Risk What You've Won (RWYW)

```
$v^* \leftarrow$ value of the game to player $i$
$k^1 \leftarrow 0$
for $t = 1$ to $T$ do
    $\pi^t \leftarrow \text{argmax}_{\pi \in \text{SAFE}(k^t)} M(\pi)$
    Play action $a_i^t$ according to $\pi^t$
    Update $M$ with opponent's actions, $a_{-i}^t$
    $k^{t+1} \leftarrow k^t + u_i(a_i^t, a_{-i}^t) - v^*$
end for
```

Proposition 10. *RWYW is not safe.*

A better approach than RWYW would be to risk the amount won so far *in expectation*. Ideally we would like to do the expectation over both our randomization and our opponent's, but this is not possible in general since we only observe the opponent's action, not his full strategy. However, it would be possible to do the expectation only over our randomization. It turns out that we can indeed achieve safety using this procedure, which we call RWYWE. Pseudocode is given in Algorithm 3. Here $u_i(\pi_i^t, a_{-i}^t)$ denotes our expected payoff of playing our mixed strategy π_i^t against the opponent's observed action a_{-i}^t.

Algorithm 3 Risk What You've Won in Expectation (RWYWE)

```
$v^* \leftarrow$ value of the game to player $i$
$k^1 \leftarrow 0$
for $t = 1$ to $T$ do
    $\pi^t \leftarrow \text{argmax}_{\pi \in \text{SAFE}(k^t)} M(\pi)$
    Play action $a_i^t$ according to $\pi^t$
    The opponent plays action $a_{-i}^t$ according to unobserved distribution $\pi_{-i}^t$
    Update $M$ with opponent's actions, $a_{-i}^t$
    $k^{t+1} \leftarrow k^t + u_i(\pi_i^t, a_{-i}^t) - v^*$
end for
```

Proposition 11. *RWYWE is safe.*

We also propose several other safe exploitative algorithms, which alter between playing a best response and an equilibrium. These algorithms are less aggressive than RWYWE, because they only play a (non-equilibrium) exploitative strategy when enough gifts have been accrued to risk playing a full best response (while RWYWE will potentially deviate from the stage-game equilibrium once any gifts have been accrued). In Proposition 12, we provide a full characterization of all safe strategies. We also present analogous algorithms and characterization results in sequential games of perfect and imperfect information, where we make pessimistic assumptions that the opponent is playing a best response in unobserved portions of the game.

Definition 3. *An algorithm for selecting strategies is* expected-profit-safe *if it satisfies the rule*

$$\pi^t \in \textit{SAFE}(k^t)$$

at each time step t from 1 to T, where initially $k^1 = 0$ and k is updated using the rule

$$k^{t+1} \leftarrow k^t + u_i(\pi^t, a^t_{-i}) - v^*.$$

Proposition 12. *A strategy π (for the full game, not the stage game) is safe if and only if it is expected-profit-safe.*

We showed that our safe exploitation algorithms significantly outperformed playing the best Nash equilibrium against several opponent classes in Kuhn poker, a sequential game of imperfect information. Against strong dynamic opponents, our algorithms significantly outperformed playing a full best response, which obtained a payoff below the value of the game. Overall, our most aggressive algorithm, RWYWE, performed best.

Table 4: Win rate in \$/hand of the five algorithms against opponents from each class. The $\pm$ given is the 95% confidence interval.

	Opponent			
	Random	Sophisticated static	Dynamic	Equilibrium
RWYWE	0.3636 $\pm$ 0.0004	-0.0110 $\pm$ 0.0004	-0.02043 $\pm$ 0.00044	-0.0556 $\pm$ 0.0004
BEFEWP	0.3553 $\pm$ 0.0004	-0.0115 $\pm$ 0.0004	-0.02138 $\pm$ 0.00045	-0.0556 $\pm$ 0.0004
BEFFE	0.1995 $\pm$ 0.0004	-0.0131 $\pm$ 0.0004	-0.03972 $\pm$ 0.00044	-0.0556 $\pm$ 0.0004
Best Nash	0.1450 $\pm$ 0.0004	-0.0148 $\pm$ 0.0004	-0.03522 $\pm$ 0.00044	-0.0556 $\pm$ 0.0004
Best response	**0.4700 $\pm$ 0.0004**	**0.0548 $\pm$ 0.0004**	**-0.12094 $\pm$ 0.00039**	**-0.0556 $\pm$ 0.0004**

In some matches, RWYWE steadily accumulates gifts along the way, and k^t increases throughout the match. An example of the graph of profit and k^t for one such opponent is given in Figure 10. In this situation, the opponent is frequently giving us gifts, and we quickly start playing (and continue to play) a full best response according to our opponent model. In other matches, k^t remains very close to 0 throughout the match, despite the fact that profits are steadily increasing; one such example is given in Figure 11. Against this opponent, we are frequently playing an equilibrium or an ϵ-safe best response for some small ϵ, and only occasionally playing a full best response.

4.3 New algorithm for exploiting opponents in large imperfect-information games using game decomposition and real-time equilibrium computation

We also propose a new algorithm for opponent exploitation in large imperfect-information games that addresses limitations of DBBR, which we intend to apply to no-limit Texas Hold'em. The new algorithm will perform opponent modeling in the preflop betting round only. To compute the opponent model, we will consider both DBBR's approach and a new constrained clustering algorithm. For both approaches, we will take into account hands that go to showdown by fixing the opponent to take the observed action with those private cards (DBBR does not take showdowns into account). Once the preflop opponent model has been constructed, we will compute an approximate equilibrium in the flop 'midgame' using the approach described in Section 3.1. Note that our endgame (and midgame) solver takes any distribution of private hands as input; in particular, we can input the precomputed approximate equilibrium for our strategy, and an

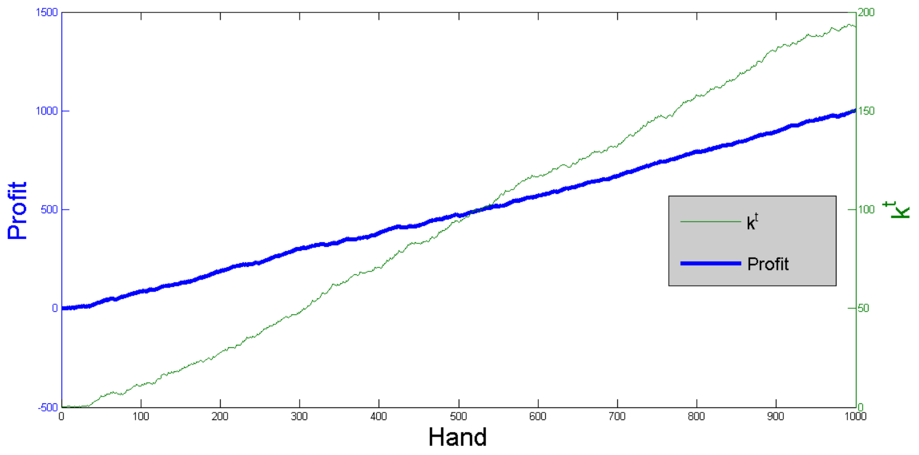


Figure 10: Profit and k^t over the course of a match of RWYWE against a random opponent. Profits are denoted by the thick blue line using the left Y axis, while k^t is denoted by the thin green line and the right Y axis. Against this opponent, both k^t and profits steadily increase.

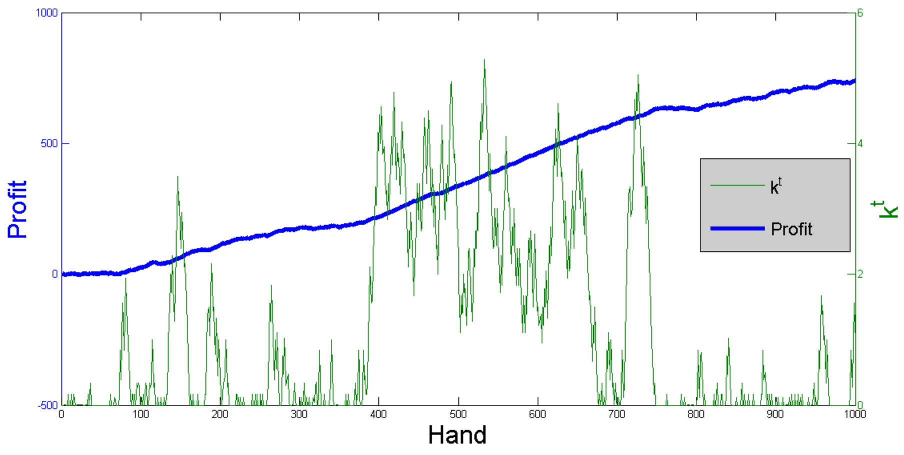


Figure 11: Profit and k^t over the course of a match of RWYWE against a random opponent. Profits are denoted by the thick blue line using the left Y axis, while k^t is denoted by the thin green line and the right Y axis. Against this opponent, k^t stays relatively close to 0 throughout the match, while profit steadily increases.

opponent model for the opponent's strategy (as opposed to the approximate equilibrium for both players). We can then propagate the new distributions induced by the flop midgame solver as inputs to the relevant turn midgame, and similarly to the river endgame. For the preflop round, we can either just play the equilibrium strategy, or we can compute an ϵ-safe best response to the opponent's strategy just within the preflop game. While it is generally intractable to compute ϵ-safe best responses in real time, we can do it in this situation because we are solving smaller games.

This new algorithm takes into account several limitations of DBBR. First, it accounts for situations when the opponent's private information is either observed or unobserved after the hand. Second, it is significantly less exploitable, since it is playing an equilibrium of a modified game where the opponent's hand distribution is fixed, rather than playing a full best response. Third, it takes advantage of all of the benefits of endgame solving described in Section 3.1. In particular, it allows for much better abstractions in the relevant portions of the game tree. By contrast, DBBR must use very coarse abstractions so that full best responses can be computed. And finally, we propose to only do the opponent modeling component for the preflop round. This has several advantages. First, we have many more observations in the preflop round than in later rounds. And second, DBBR only uses aggregate statistics for the later rounds, since it uses public history sets, which abstract away relevant information like the public cards. For example, an opponent may bet on 80% of flops overall, but on certain flops he will bet 100% of the time and on others he will only bet 10%. Using one aggregate statistic is probably too simplistic to be useful against any relatively strong opponent. We expect our new algorithm to be able to successfully exploit even strong no-limit Texas Hold'em opponents.

5 Summary and timeline

5.1 Completed work

- Computing equilibria in multiplayer stochastic imperfect-information games [11, 12].
- Computing equilibria by incorporating qualitative models [13, 14].
- Strategy purification and thresholding [20].
- Action translation [18].
- Tartanian5 agent for no-limit Texas Hold'em [17].
- Endgame solving [19].
- Algorithm for opponent exploitation in large imperfect-information games [15].
- Safe opponent exploitation [16]. Journal version in submission to TEAC.

5.2 Work to be done

- Improvements for qualitative model project: implement new polynomial-time algorithm and try to characterize the situations in which the different models are used in limit Texas Hold'em.
- New algorithm for information abstraction based on a new algorithms for computing earth-mover's distance between histograms of unordered clusters.
- New algorithm for information abstraction that will allow MCCFR to parallelize on ccNUMA architecture.

- Gain a better theoretical understanding of purification and thresholding by providing formal proofs of Observations 1 and 2, and generalizing results to games of arbitrary size.
- Apply machine learning algorithms to extract human-understandable knowledge from the computed strategy files.
- Additional enhancements and experiments for the endgame solver.
- Extending the endgame solver to solve intermediate portions of the game.
- New algorithm for solving large imperfect-information games by creating an endgame database and integrating it with MCCFR.
- New algorithm for opponent exploitation that utilizes the endgame solver.

5.3 Timeline

- 9/2013: Thesis proposal
- 8/2013–10/2013: Improve endgame solving paper for submission to AAMAS.
- 8/2013–12/2013: Implement new information abstraction algorithms on no-limit Texas Hold'em.
- 12/2013–6/2014: Implement midgame solver, new opponent exploitation algorithm, and new game-solving algorithm that uses an endgame database. Work on no-limit Texas Hold'em agents for the Annual Computer Poker Competition.
- 6/2014–8/2014: Work on approaches for extracting human-understandable knowledge from strategies.
- 6/2014–12/2014: Work on new conference and journal papers.
- 12/2014: Thesis defense

References

[1] Nick Abou Risk and Duane Szafron. Using counterfactual regret minimization to create competitive multiplayer poker agents. In *International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, 2010.

[2] Jerrod Ankenman and Bill Chen. *The Mathematics of Poker*. ConJelCo LLC, 2006.

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

[4] Nolan Bard, Michael Johanson, Neil Burch, and Michael Bowling. Online implicit agent modelling. In *International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, 2013.

[5] 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.

[6] Ronen Brafman and Moshe Tennenholtz. Learning to coordinate efficiently: A model-based approach. *Journal of Artificial Intelligence Research*, 19:11–23, 2003.

[7] George W. Brown. Iterative solutions of games by fictitious play. In Tjalling C. Koopmans, editor, *Activity Analysis of Production and Allocation*, pages 374–376. John Wiley & Sons, 1951.

[8] Xi Chen and Xiaotie Deng. Settling the complexity of 2-player Nash equilibrium. In *Proceedings of the Annual Symposium on Foundations of Computer Science (FOCS)*, 2006.

[9] George Dantzig. A proof of the equivalence of the programming problem and the game problem. In Tjalling Koopmans, editor, *Activity Analysis of Production and Allocation*, pages 330–335. John Wiley & Sons, 1951.

[10] Constantinos Daskalakis, Paul Goldberg, and Christos Papadimitriou. The complexity of computing a Nash equilibrium. In *Proceedings of the Annual Symposium on Theory of Computing (STOC)*, 2006.

[11] Sam Ganzfried and Tuomas Sandholm. Computing an approximate jam/fold equilibrium for 3-player no-limit Texas Hold'em tournaments. In *International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, 2008.

[12] Sam Ganzfried and Tuomas Sandholm. Computing equilibria in multiplayer stochastic games of imperfect information. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI)*, 2009.

[13] Sam Ganzfried and Tuomas Sandholm. Computing equilibria by incorporating qualitative models. In *International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, 2010.

[14] Sam Ganzfried and Tuomas Sandholm. Computing equilibria by incorporating qualitative models. Technical Report CMU-CS-10-105, Computer Science Department, Carnegie-Mellon University, Pittsburgh, PA, 2010.

[15] Sam Ganzfried and Tuomas Sandholm. Game theory-based opponent modeling in large imperfect-information games. In *International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, 2011.

[16] Sam Ganzfried and Tuomas Sandholm. Safe opponent exploitation. In *ACM Conference on Electronic Commerce (EC)*, 2012.

[17] Sam Ganzfried and Tuomas Sandholm. Tartanian5: A heads-up no-limit Texas Hold'em poker-playing program. In *Computer Poker Symposium at the National Conference on Artificial Intelligence (AAAI)*, 2012.

[18] 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.

[19] Sam Ganzfried and Tuomas Sandholm. Improving performance in imperfect-information games with large state and action spaces by solving endgames. In *Computer Poker and Imperfect Information Workshop at the National Conference on Artificial Intelligence (AAAI)*, 2013.

[20] Sam Ganzfried, Tuomas Sandholm, and Kevin Waugh. Strategy purification and thresholding: Effective non-equilibrium approaches for playing large games. In *International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, 2012.

[21] Richard Gibson. Regret minimization in non-zero-sum games with applications to building champion multiplayer computer poker agents. *ArXiv e-prints*, April 2013.

[22] Richard Gibson, Neil Burch, Marc Lanctot, and Duane Szafron. Efficient Monte Carlo counterfactual regret minimization in games with many player actions. In *Proceedings of the Annual Conference on Neural Information Processing Systems (NIPS)*, 2012.

[23] Richard Gibson, Marc Lanctot, Neil Burch, Duane Szafron, and Michael Bowling. Generalized sampling and variance in counterfactual regret minimization. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, 2012.

[24] Andrew Gilpin, Javier Peña, and Tuomas Sandholm. First-order algorithm with $\mathcal{O}(\ln(1/\epsilon))$ convergence for ϵ-equilibrium in two-person zero-sum games. *Mathematical Programming*, 133(1–2):279–298, 2012. Conference version appeared in AAAI-08.

[25] 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)*, 2006.

[26] Andrew Gilpin and Tuomas Sandholm. Better automated abstraction techniques for imperfect information games, with application to Texas Hold'em poker. In *International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, pages 1168–1175, 2007.

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

[28] Andrew Gilpin, Tuomas Sandholm, and Troels Bjerre Sørensen. 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)*, 2007.

[29] Andrew Gilpin, Tuomas Sandholm, and Troels Bjerre Sørensen. A heads-up no-limit Texas Hold'em poker player: Discretized betting models and automatically generated equilibrium-finding programs. In *International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, 2008.

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

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

[32] Samid Hoda, Andrew Gilpin, Javier Peña, and Tuomas Sandholm. Smoothing techniques for computing Nash equilibria of sequential games. *Mathematics of Operations Research*, 35(2):494–512, 2010.

[33] Junling Hu and Michael P. Wellman. Nash Q-learning for general-sum stochastic games. *Journal of Machine Learning Research*, 4:1039–1069, 2003.

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

[35] Michael Johanson, Nolan Bard, Neil Burch, and Michael Bowling. Finding optimal abstract strategies in extensive-form games. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, 2012.

[36] Michael Johanson, Nolan Bard, Marc Lanctot, Richard Gibson, and Michael Bowling. Efficient Nash equilibrium approximation through Monte Carlo counterfactual regret minimization. In *International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, 2012.

[37] Michael Johanson and Michael Bowling. Data biased robust counter strategies. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2009.

[38] 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.

[39] Michael Johanson, Martin Zinkevich, and Michael Bowling. Computing robust counter-strategies. In *Proceedings of the Annual Conference on Neural Information Processing Systems (NIPS)*, 2007.

[40] Michael Kearns, Yishay Mansour, and Satinder Singh. Fast planning is stochastic games. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI)*, 2000.

[41] Daphne Koller, Nimrod Megiddo, and Bernhard von Stengel. Fast algorithms for finding randomized strategies in game trees. In *Proceedings of the 26th ACM Symposium on Theory of Computing (STOC)*, 1994.

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

[43] Marc Lanctot, Kevin Waugh, Martin Zinkevich, and Michael Bowling. Monte Carlo sampling for regret minimization in extensive games. In *Proceedings of the Annual Conference on Neural Information Processing Systems (NIPS)*, pages 1078–1086, 2009.

[44] Michael Littman. Friend-or-foe Q-learning in general-sum Markov games. In *International Conference on Machine Learning (ICML)*, pages 322–328, 2001.

[45] Peter Bro Miltersen and Troels Bjerre Sørensen. Fast algorithms for finding proper strategies in game trees. In *Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2008.

[46] Peter Bro Miltersen and Troels Bjerre Sørensen. Computing a quasi-perfect equilibrium of a two-player game. *Economic Theory*, 42(1):175–192, 2010.

[47] John Nash. Non-cooperative games. *Annals of Mathematics*, 54:289–295, 1951.

[48] Martin L Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, 2005.

[49] Jonathan Rubin and Ian Watson. Computer poker: A review. *Artificial Intelligence*, 175(5–6):958–987, 2011.

[50] Tuomas Sandholm. Perspectives on multiagent learning. *Artificial Intelligence*, 171:382–391, 2007.

[51] 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.

[52] David Schnizlein, Michael Bowling, and Duane Szafron. Probabilistic state translation in extensive games with large action sets. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI)*, 2009.

[53] Satinder P Singh, Vishal Soni, and Michael P Wellman. Computing approximate Bayes-Nash equilibria in tree-games of incomplete information. In *Proceedings of the ACM Conference on Electronic Commerce (ACM-EC)*, pages 81–90, New York, NY, 2004. ACM.

[54] Noah D. Stein, Asuman Ozdaglar, and Pablo A. Parillo. Separable and low-rank continuous games. *International Journal of Game Theory*, 37(4):475–504, 2008.

[55] John von Neumann. Zur Theorie der Gesellschaftsspiele. *Mathematische Annalen*, 100:295–320, 1928.

[56] John von Neumann and Oskar Morgenstern. *Theory of Games and Economic Behavior*. Princeton University Press, 1947.

[57] Yevgeniy Vorobeychik and Michael Wellman. Stochastic search methods for Nash equilibrium approximation in simulation-based games. In *International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, Estoril, Portugal, 2008.

[58] Xiaofeng Wang and Tuomas Sandholm. Reinforcement learning to play an optimal Nash equilibrium in team Markov games. In *Proceedings of the Annual Conference on Neural Information Processing Systems (NIPS)*, 2002.

[59] Kevin Waugh. Abstraction in large extensive games. Master's thesis, University of Alberta, 2009.

[60] Kevin Waugh, Martin Zinkevich, Michael Johanson, Morgan Kan, David Schnizlein, and Michael Bowling. A practical use of imperfect recall. In *Symposium on Abstraction, Reformulation and Approximation (SARA)*, 2009.

[61] 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.