Strategy Purification and Thresholding: Effective Non-Equilibrium Approaches for Playing Large Games*

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

ABSTRACT

There has been significant recent interest in computing effective strategies for playing large imperfect-information games. Much prior work involves computing an approximate equilibrium strategy in a smaller abstract game, then playing this strategy in the full game (with the hope that it also well approximates an equilibrium in the full game). In this paper, we present a family of modifications to this approach that work by constructing non-equilibrium strategies in the abstract game, which are then played in the full game. Our new procedures, called *purification* and *thresholding*, modify the action probabilities of an abstract equilibrium by preferring the higher-probability actions. Using a variety of domains, we show that these approaches lead to significantly stronger play than the standard equilibrium approach. As one example, our program that uses purification came in first place in the two-player no-limit Texas Hold'em total bankroll division of the 2010 Annual Computer Poker Competition. Surprisingly, we also show that purification significantly improves performance (against the full equilibrium strategy) in random 4×4 matrix games using random 3×3 abstractions. We present several additional results (both theoretical and empirical). Overall, one can view these approaches as ways of achieving robustness against overfitting one's strategy to one's lossy abstraction. Perhaps surprisingly, the performance gains do not necessarily come at the expense of worst-case exploitability.

Categories and Subject Descriptors

I.2.11 [**Distributed Artificial Intelligence**]: Multiagent Systems; J.4 [**Social and Behavioral Sciences**]: Economics

General Terms

Algorithms, Economics, Theory

Keywords

Game theory, game playing, large games, abstraction, reverse mapping, non-equilibrium approaches, purification

1. INTRODUCTION

Developing effective strategies for agents in multiagent systems is an important and challenging problem. It has received significant attention in recent years from several different communities—in part due to the competitions held at top conferences (e.g, the computer poker, robo-soccer, and trading agent competitions). As many domains are so large that solving them directly (i.e., computing a Nash equilibrium or a solution according to some other relevant solution concept) is computationally infeasible, some amount of approximation is necessary to produce agents.

Specifically, significant work has been done on computing approximate game-theory-based strategies in large imperfect-information games. This work typically follows a three-step approach, which is depicted in Figure 1. First, an *abstraction algorithm* is run on the original game G to construct a smaller game G' which is strategically similar to G [1, 2, 3, 11]. Second, an *equilibrium-finding algorithm* is run on G' to compute an ϵ-equilibrium σ' [6, 13]. Third, a *reverse mapping* is applied to σ' to compute an approximate equilibrium σ in the full game G [5, 10]. While most prior work has focused on the first two steps of this approach, in this paper we focus on the third. In particular, we propose first mapping the abstract approximate-equilibrium strategy profile to a *non-equilibrium* strategy profile in the abstract game, which we then map to a strategy profile in the full game.

Almost all prior work has used the trivial reverse mapping in which σ is the straightforward projection of σ' into G. In other words, once the abstract game is solved, its solution is just played directly in the full game. In this paper, we show that applying more sophisticated reverse mappings can lead to significant performance improvements—even if they produce strategy profiles that are no longer equilibria in the abstract game.

One of the key ideas that motivated our approach is that the exact action probabilities of a mixed strategy equilibrium in an abstraction can exemplify overfitting to the particular abstraction used. (Our results confirm this.) Ideally, we would like to extrapolate general principles from the strategy rather than just use values that were finely

*This material is based upon work supported by the National Science Foundation under grants IIS-0964579, IIS-0905390, and CCF-1101668. We also acknowledge Intel Corporation and IBM for their machine gifts. Early versions of this paper appeared as "Strategy Purification" in the National Conference on Artificial Intelligence (AAAI) Workshop on Applied Adversarial Reasoning and Risk Modeling, 2011, and as a 2-page abstract in the International Conference on Autonomous Agents and Multiagent Systems (AAMAS), 2011.

Appears in: *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012)*, Conitzer, Winikoff, Padgham, and van der Hoek (eds.), 4-8 June 2012, Valencia, Spain.

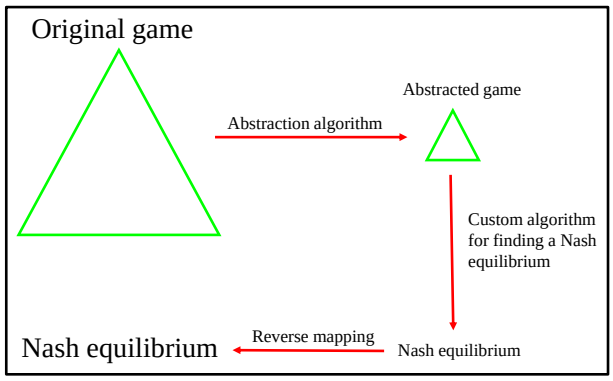

Figure 1: General approach for solving large games.

tuned for a specific abstraction. This is akin to the classic example from machine learning, where we would prefer a degree-one polynomial that fits the training data quite well to a degree-hundred polynomial that may fit it slightly better. (Subsequent to the appearance of the earlier versions of our paper, others have also shown that overfitting strategies to a particular abstraction is a very significant and real problem in large imperfect-information games [7].)

We present a 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. Our new procedures, called *purification* and *thresholding*, modify the action probabilities of an abstract equilibrium by placing a preference on the higher-probability actions. The main intuition behind our algorithms is that we should ignore actions that are played with small probability in the abstract equilibrium, as they are likely due to abstraction coarseness, overfitting, or failure of the equilibrium-finding algorithm to fully converge.

Using a variety of experimental domains, we show that our new approach leads to significantly stronger play than the standard abstraction/equilibrium approach. For example, our program that uses purification won the two-player no-limit Texas Hold'em total bankroll division of the 2010 Annual Computer Poker Competition (ACPC), held at AAAI. Surprisingly, we also show that purification significantly improves performance (against the full equilibrium strategy) in random 4×4 matrix games using random 3×3 abstractions. We present additional results (both theoretical and empirical), including: worst-case theoretical results, empirical and theoretical results on specific support properties for which purification helps in matrix games, and experimental results in well-studied large imperfect-information games (Leduc Hold'em and Texas Hold'em).

2. GAME THEORY BACKGROUND

In this section, we briefly review relevant definitions and prior results from game theory and game solving.

2.1 Strategic-form games

The most basic game representation, and the standard representation for simultaneous-move games, is the *strategic form*. A *strategic-form game* (aka matrix game) consists of a finite set of players N, a space 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 *strategy profiles*—vectors of pure strategies, one for each player.

The set of *mixed strategies* of player i is the space of probability distributions over his pure strategy space S_i. We will denote this space by Σ_i. Define the *support* of a mixed strategy to be the set of pure strategies played with nonzero probability. If the sum of the payoffs of all players equals zero at every strategy profile, then the game is called *zero sum*. In this paper, we will be primarily concerned with two-player zero-sum games; we will show that the new approaches lead to performance improvements even in this class of games where the equilibrium approach should be at its best. If the players are following strategy profile σ, we let σ_{-i} denote the strategy taken by player i's opponent, and we let Σ_{-i} denote the opponent's entire mixed strategy space.

2.2 Extensive-form games

An *extensive-form* game is a general model of multiagent decision making with potentially sequential and simultaneous actions and imperfect information. As with perfect-information games, 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.

2.3 Nash equilibria

Player i's *best response* to σ_{-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. An *ϵ-equilibrium* is a strategy profile in which each player achieves a payoff of within ϵ of his best response.

In two-player zero-sum games, we have the following result which is known as the *minimax theorem*:

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

We refer to v^* as the *value* of the game to player 1. Sometimes we will write v_i as the value of the game to player i. It is worth noting that in two-player zero-sum games, any equilibrium strategy for a player will guarantee an expected payoff of at least the value of the game to that player.

All finite games have at least one Nash equilibrium. In two-player zero-sum strategic-form games, a Nash equilibrium can be found efficiently by linear programming. In the case of zero-sum extensive-form games with perfect recall, there are efficient techniques for finding an ϵ-equilibrium, such as linear programming [8], generalizations of the excessive gap technique [6], and counterfactual regret minimization [13]. The latter two algorithms scale to games with approximately 10^{12} game tree states, while the best current

general-purpose linear programming technique (CPLEX's barrier method) scales to games with around 10^8 states.

2.4 Abstraction

Despite the tremendous progress in equilibrium-finding in recent years, many interesting real-world games are so large that even the best algorithms have no hope of computing an equilibrium directly. The standard approach of dealing with this is to apply an *abstraction* algorithm, which constructs a smaller game that is similar to the original game; then the smaller game is solved, and its solution is mapped to a strategy profile in the original game. The approach has been applied to two-player Texas Hold'em poker, first with a manually generated abstraction [1], and now with abstraction algorithms [2]. Many abstraction algorithms work by coarsening the moves of chance, collapsing several information sets of the original game into single information sets of the abstracted game.

The game tree of two-player no-limit Texas Hold'em has about 10^{71} states (while that of two-player limit Texas Hold'em has about 10^{18} states); so significant abstraction is necessary.

3. PURIFICATION AND THRESHOLDING

Suppose we are playing a game Λ that is too large to solve directly. As described in Section 2.4, the standard approach would be to construct an abstract game Λ', compute an equilibrium σ' of Λ', then play the strategy profile σ induced by σ' in the full game Λ.

One possible problem with this approach is that the specific strategy profile σ' might be very finely tuned for the abstract game Λ', and it could perform arbitrarily poorly in the full game (see the results in Section 5). Ideally we would like to extrapolate the important features from σ' that will generalize to the full game and avoid playing a strategy that is overfit to the particular abstraction. This is one of the key motivations for our new approaches, *purification* and *thresholding*.

3.1 Purification

Let σ_i be a mixed strategy for player i in a strategic-form game, and let $S = \arg\max_j \sigma_i(j)$, where j ranges over all of player i's pure strategies. Then we define the *purification* $\text{pur}(\sigma_i)$ of σ_i as follows:

$$\text{pur}(\sigma_i)(j) = \begin{cases} 0 & : \quad j \notin S \\ \frac{1}{|S|} & : \quad j \in S \end{cases}$$

If 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. Thus the purification will usually be a pure strategy, and will only be a mixed strategy in degenerate special cases when several pure strategies are played with identical probabilities.

If σ_i is a behavioral strategy in an extensive-form game, we define the purification similarly; at each information set I, $\text{pur}(\sigma_i)$ will play the purification of σ_i at I.

3.2 Thresholding

The effects of purification can be quite extreme in some situations. For example, if σ_i plays action a with probability 0.51 and action b with probability 0.49, then b will never be played after purification. We also consider a more relaxed approach, called *thresholding*, that only eliminates actions below a prescribed ϵ to help alleviate this concern.

Thresholding works by setting all actions that have weight below ϵ to 0, then renormalizing the action probabilities. Pseudocode is given below in Algorithm 1. One intuitive interpretation of thresholding is that actions with probability below ϵ may just have been given positive probability due to noise from the abstraction (or because an equilibrium-finding algorithm had not yet taken those probabilities all the way to zero), and really should not be played in the full game. Additionally, low probability actions are often played primarily to protect a player from being exploited, and this may be an overstated concern against realistic opponents (as discussed further in Section 4.2).

Algorithm 1 Threshold(σ_i, ϵ)

```
for j = 1 to |S| do
  if σ_i(j) < ε then
    σ_i(j) ← 0
  end if
end for
normalize(σ_i)
return σ_i
```

4. EVALUATION METRICS

In recent years, several different metrics have been used to evaluate strategies in large games.

4.1 Empirical performance

The first metric, which is perhaps the most meaningful, is *empirical performance* against other realistic strategies. For example, in the ACPC, programs submitted from researchers and hobbyists from all over the world compete against one another. Empirical performance is the metric we will be using in Section 8 when we assess our performance in Texas Hold'em.

4.2 Worst-case exploitability

The worst-case *exploitability* of player i's strategy σ_i is the difference between the value of the game to player i and the payoff when the opponent plays his best response to σ_i (aka his *nemesis strategy*). Formally it is defined as follows:

$$\text{expl}(\sigma_i) = v_i - \min_{\sigma_{-i} \in \Sigma_{-i}} u_i(\sigma_i, \sigma_{-i}).$$

Worst-case exploitability has recently been used to assess strategies in several variants of poker [4, 7, 12].

Any equilibrium has zero exploitability, since it receives payoff v_i against its nemesis. So if our goal were to approximate an equilibrium of the full game, worst-case exploitability would be a good metric to use, since it approaches zero as the strategy approaches equilibrium.

Unfortunately, the worst-case exploitability metric has several drawbacks. First, it cannot be computed in very large games (though very recent advancements have made it possible to compute full best responses offline in two-player limit Texas Hold'em, which has about 10^{18} game states [7], and we will be leveraging that algorithm in our experiments).

Second, exploitability is a worst-case metric that assumes the opponent is able to *optimally* exploit us in the full game (i.e., he knows our full strategy and is able to efficiently compute a full best response in real time). In fact, it is quite common in very large games for agents to simply play static, fixed strategies the entire time, since the number of interactions is generally tiny compared to the size of the game, and it is usually quite difficult to learn to effectively exploit opponents online. For example, in recent computer poker competitions, almost all submitted programs simply play a fixed strategy. In the 2010 ACPC, many of the entrants attached summaries describing their algorithm. Of the 17 bots for which summaries were included, 15 played fixed strategies, while only 2 included some element of attempted exploitation. If the opponents are just playing a fixed strategy and not trying to exploit us, then worst-case exploitability is too pessimistic of an evaluation metric. Furthermore, if the opponents have computational limitations and use abstractions, then they will not be able to fully exploit us in the full game.

4.3 Performance against full equilibrium

In this paper, we will also evaluate strategies based on performance against equilibrium in the full game. The intuition behind this metric is that in many large two-player zero-sum games, the opponents are simply playing fixed strategies that attempt to approximate an equilibrium of the full game (using some abstraction). For example, most entrants in the ACPC do this. Against such static opponents, worst-case exploitability is not very significant, as the agents are not generally adapting to exploit us.

This metric, like worst-case exploitability, is not feasible to apply on very large games. However, we can still apply it to smaller games as a means of comparing different solution techniques. In particular, we will use this metric in Sections 6 and 7 when presenting our experimental results on random matrix games and Leduc Hold'em. This metric has similarly been used on solvable problem sizes in the past to compare abstraction algorithms [4].

5. THEORY: SELECTIVE SUPERIORITY

So which approach is best: purification, thresholding, or the standard abstraction/equilibrium approach? It turns out that using the performance against full equilibrium metric, there exist games for which each technique can outperform each other. Thus, from a worst-case perspective, not much can be said in terms of comparing the approaches.

Proposition 1 shows that, for *any* equilibrium-finding algorithm, there exists a game and an abstraction such that purification does arbitrarily better than the standard approach.

PROPOSITION 1. *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.

PROOF. Consider the game in Figure 2. Let Λ denote the full game, and let Λ' denote the abstraction in which player 2 (the column player) is restricted to only playing L or M,

	L	M	R
U	2	0	$-3k-1$
D	0	1	-1

Figure 2: Two-player zero-sum game used in the proof of Proposition 1.

but the row player's strategy space remains the same. Then Λ' has a unique equilibrium in which player 1 plays U with probability $\frac{1}{3}$, and player 2 plays L with probability $\frac{1}{3}$. Since this is the unique equilibrium, it must be the one output by algorithm A'. Note that player 1's purification $\text{pur}(\sigma'_1)$ of σ' is the pure strategy D.

Note that in the full game Λ, the unique equilibrium is (D,R), which we denote by σ. As before, since this equilibrium is unique it must be the one output by algorithm A. Then we have

$$\begin{aligned} u_1(\sigma'_1, \sigma_2) = \frac{1}{3}(-3k-1) + \frac{2}{3}(-1) &= -k-1 \\ u_1(\text{pur}(\sigma'_1), \sigma_2) &= -1. \end{aligned}$$

So $u_1(\sigma'_1, \sigma_2) + k = -1$, and therefore

$$u_1(\text{pur}(\sigma'_1), \sigma_2) = u_1(\sigma'_1, \sigma_2) + k.$$

□

We can similarly show that purification can also do arbitrarily worse against the full equilibrium than standard 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.

6. RANDOM MATRIX GAMES

The first set of experiments we conduct to demonstrate the power of purification is on random matrix games. This is perhaps the most fundamental and easy to analyze class of games, and is a natural starting point when analyzing new algorithms.

6.1 Evaluation methodology

We study random 4×4 two-player zero-sum matrix games with payoffs drawn uniformly at random from [-1,1]. We repeatedly generated random games and analyzed them using the following procedure. First, we computed an equilibrium of the full 4×4 game Λ; denote this strategy profile by σ^F. Next, we constructed an abstraction Λ' of Λ by ignoring the final row and column of Λ. In essence, Λ' is a naïve, random abstraction of Λ, since there is nothing special about the final row or column. As in Λ, we computed an equilibrium σ^A of Λ'. We then compared $u_1(\sigma_1^A, \sigma_2^F)$ to $u_1(\text{pur}(\sigma_1^A), \sigma_2^F)$ to determine the effect of purification on performance of the abstract equilibrium strategy for player 1 against the full equilibrium strategy of player 2.

To solve the full and abstract games, we used two different procedures. For our first set of experiments comparing the overall performance of purified vs. unpurified abstract equilibrium strategies, we used a standard algorithm involving solving a single linear program [8]. For our results on supports, we used a custom support enumeration algorithm (similar to the approach of Porter et al. [9]). We note that it

is possible that the specific algorithm used may have a significant effect on the results (i.e., certain algorithms may be more likely to select equilibria with specific properties when several equilibria exist).

6.2 Experimental results and theory

In our experiments on 4×4 random games, we performed 1.5 million trials; the results are given in Table 1. The first row gives the average value of $u_1(\text{pur}(\sigma_1^A), \sigma_2^F)$ over all trials, while the second row gives the average value of $u_1(\sigma_1^A, \sigma_2^F)$. We conclude that purified abstraction outperforms the standard unpurified abstraction approach using 95% confidence intervals.

The next three rows of Table 1 report the number of trials for which purification led to an increased, decreased, or unchanged payoff of the abstract equilibrium strategy of player 1 against the full equilibrium strategy of player 2. While purification clearly improved performance more often than it hurt performance (17.44% vs. 11.48%), for the overwhelming majority of cases it led to no change in performance (71.08%). In particular, Proposition 2 gives two general sets of conditions under which purification leads to no change in performance.

PROPOSITION 2. *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. *$support(\sigma_1^A) \subseteq support(\sigma_1^F)$*

PROOF. If the first condition is met, then $\text{pur}(\sigma_1^A) = \sigma_1^A$ and we are done. Now suppose the second condition is true and let $s, t \in support(\sigma_1^A)$ be arbitrary. This implies that $s, t \in support(\sigma_1^F)$ as well, which means that $u_1(s, \sigma_2^F) = u_1(t, \sigma_2^F)$, since a player is indifferent between all pure strategies in his support at an equilibrium. Since s and t were arbitrary, player 1 is also indifferent between all strategies in $support(\sigma_1^A)$ when player 2 plays σ_2^F. Since purification will just select one strategy in $support(\sigma_1^A)$, we are done. □

To understand our results further, we investigated whether they would vary for different supports of σ^F. In particular, we kept separate tallies of the performance of $\text{pur}(\sigma_1^A)$ and σ_1^A for each support of σ^F. We observed that $\text{pur}(\sigma_1^A)$ outperformed σ_1^A on many of the supports, while they performed equally on some (and σ_1^A did not outperform $\text{pur}(\sigma_1^A)$ on any). These results are all statistically significant using 95% confidence intervals. A summary of the results from these experiments is given in Observation 1.

OBSERVATION 1. *In random 4×4 matrix games using 3×3 abstractions, $pur(\sigma_1^A)$ 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 $pur(\sigma_1^A)$ 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.*

To interpret Observation 1, consider the following example. Suppose the support for player 1 includes his first three pure strategies, while the support for player 2 includes his final three pure strategies; denote this support profile by S^*. Now consider the set U of all games for which our equilibrium-finding algorithm outputs an equilibrium profile σ^F with support profile S^*. Since S^* does not satisfy either condition of Observation 1, this means that, in expectation over the set of all games in U,

$$u_1(\text{pur}(\sigma_1^A), \sigma_2^F) > u_1(\sigma_1^A, \sigma_2^F)$$

(i.e., purification improves the performance of the abstracted equilibrium strategy of player 1 against the full equilibrium strategy of player 2).

We find it interesting that there is such a clear pattern in the support structures for which $\text{pur}(\sigma_1^A)$ outperforms σ_1^A. We obtained identical results using 3×3 games with 2×2 abstractions. We did not experiment on games larger than 4×4. While we presented experimental results that are statistically significant at the 95% confidence interval, rigorously proving that the results of Observation 1 hold even on 4×4 games with 3×3 abstractions remains a challenging open problem. Resolving this problem would shed some light on the underlying reasons behind the observed performance improvements of purification in random matrix games, which are quite surprising and unintuitive. In addition, we conjecture that a more general theoretical result will hold for general matrix games with any size, using any size random abstractions.

7. LEDUC HOLD'EM

Leduc Hold'em is a simplified poker variant that has been used in previous work to evaluate imperfect-information game-playing techniques (e.g., [12]). Leduc Hold'em is large enough that abstraction has a non-trivial impact, but unlike larger games of interest (e.g., Texas Hold'em) it is small enough that equilibrium solutions in the full game can be quickly computed. That is, Leduc Hold'em allows for rapid and thorough evaluation of game-playing techniques against a variety of opponents, including an equilibrium opponent or a best responder.

Prior to play, a deck of six cards containing two Jacks, two Queens, and two Kings is shuffled and each player is dealt a single private card. After a round of betting, a public card is dealt face up for both players to see. If either player pairs this card, he wins at showdown; otherwise the player with the higher ranked card wins. For a complete description of the betting, we refer the reader to Waugh et al. [12].

7.1 Experimental evaluation and setup

To evaluate the effects of purification and thresholding in Leduc Hold'em, we compared the performance of a number of abstract equilibrium strategies altered to varying degrees by thresholding against a single equilibrium opponent averaged over both positions. The performance of a strategy (denoted EV for expected value) was measured in millibets per hand (mb/h), where one thousand millibets is a small

$u_1(\text{pur}(\sigma_1^A), \sigma_2^F)$ (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.

bet. As the equilibrium opponent is optimal, the best obtainable performance is 0 mb/h. Note that the expected value computations in this section are exact.

We used card abstractions mimicking those produced by state-of-the-art abstraction techniques to create our abstract equilibrium strategies. Specifically, we used the five Leduc Hold'em card abstractions from prior work [12], denoted *JQK*, *JQ.K*, *J.QK*, *J.Q.K* and *full*. The abstraction *full* denotes the null abstraction (i.e., the full unabstracted game). The names of the remaining abstractions consist of groups of cards separated by periods. All cards within a group are indistinguishable to the player prior to the flop. For example, when a player using the *JQ.K* abstraction is dealt a card, he will know only if that card is a king, or if it is not a king. These abstractions can only distinguish pairs on the flop. By pairing these five card abstractions, one abstraction per player, we learned twenty four abstract equilibrium strategies using linear programming techniques. For example, the strategy *J.Q.K-JQ.K* denotes the strategy where our player of interest uses the *J.Q.K* abstraction and he assumes his opponent uses the *JQ.K* abstraction.

7.2 Purification vs. no purification

In Table 2 we present the performance of the regular and purified abstract equilibrium strategies against the equilibrium opponent. We notice that purification improves the performance in all but 5 cases. In many cases this improvement is quite substantial. In the cases where it does not help, we notice that at least one of the players is using the *JQK* card abstraction, the worst abstraction in our selection. Prior to purification, the best abstract equilibrium strategy loses at 43.8 mb/h to the equilibrium opponent. After purification, 14 of the 24 strategies perform better than the best unpurified strategy, the best of which loses at only 1.86 mb/h. That is, only five of the strategies that were improved by purification failed to surpass the best unpurified strategy.

7.3 Purification vs. thresholding

In Figure 3 we present the results of three abstract equilibrium strategies thresholded to varying degrees against the equilibrium opponent. We notice that, the higher the threshold used the better the performance tends to be. Though this trend is not monotonic, all the strategies that were improved by purification obtained their maximum performance when completely purified. Most strategies tended to improve gradually as the threshold was increased, but this was not the case for all strategies. As seen in the figure, the *JQ.K-JQ.K* strategy spikes in performance between the thresholds of 0.1 and 0.15.

From these experiments, we conclude that purification tends to improve the performance of an abstract equilibrium strategy against an unadaptive equilibrium opponent in Leduc Hold'em. Though thresholding is itself helpful, it appears that the improvement generally increases with the threshold whenever thresholding improves a strategy, with the biggest improvement achieved using full purification.

Strategy	Base EV	Purified EV	Improvement
JQ.K-J.QK	-119.46	-37.75	81.71
J.QK-full	-115.63	-41.83	73.80
J.QK-J.Q.K	-96.66	-27.35	69.31
JQ.K-J.Q.K	-96.48	-28.76	67.71
JQ.K-full	-99.30	-39.13	60.17
JQ.K-JQK	-80.14	-24.50	55.65
JQ.K-JQ.K	-59.97	-8.31	51.66
J.Q.K-J.QK	-60.28	-13.97	46.31
J.Q.K-J.Q.K	-46.23	-1.86	44.37
J.Q.K-JQ.K	-44.61	-3.85	40.76
full-JQK	-43.80	-10.95	32.85
J.QK-J.QK	-96.60	-67.42	29.18
J.QK-JQK	-95.69	-67.14	28.55
full-J.QK	-52.94	-24.55	28.39
J.QK-JQ.K	-77.86	-52.62	25.23
J.Q.K-full	-68.10	-46.43	21.66
full-JQ.K	-55.52	-36.38	19.14
full-J.Q.K	-51.14	-40.32	10.82
JQK-J.QK	-282.94	-279.44	3.50
JQK-full	-273.87	-279.99	-6.12
JQK-J.Q.K	-258.29	-279.99	-21.70
J.Q.K-JQK	-156.35	-188.00	-31.65
JQK-JQK	-386.89	-433.64	-46.75
JQK-JQ.K	-274.69	-322.41	-47.72

Table 2: Effects of purification on performance of abstract strategies against an equilibrium opponent in mb/h.

8. TEXAS HOLD'EM

In the 2010 Annual Computer Poker Competition, the CMU team (Ganzfried, Gilpin, and Sandholm) submitted bots that used both purification and thresholding to the two-player no-limit Texas Hold'em division. We present the results in Section 8.1. Next, in Section 8.2, we observe how varying the amount of thresholding used effects the exploitabilities of two bots submitted to the two-player limit Texas Hold'em division.

8.1 A champion no-limit Texas Hold'em program

The two-player no-limit competition consists of two subcompetitions with different scoring rules. In the *instant-runoff* scoring rule, each pair of entrants plays against each other, and the bot with the worst head-to-head record is eliminated. This procedure is continued until only a single bot remains. The other scoring rule is known as *total bankroll*. In this competition, all entrants play against each other and are ranked in order of their total profits. While both scoring metrics serve important purposes, the

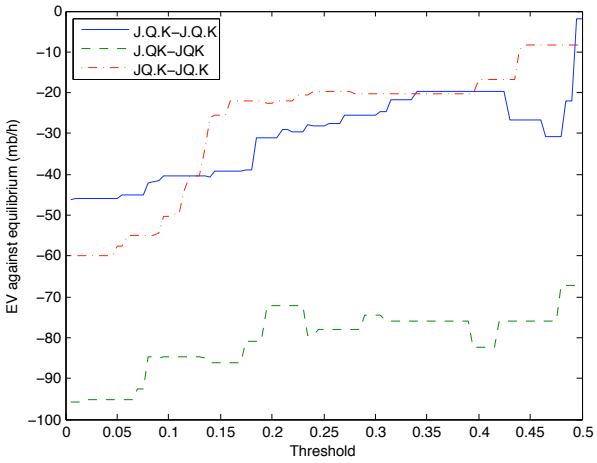

Figure 3: Effects of thresholding on performance of abstract strategies against an equilibrium opponent in mb/h.

total bankroll competition is considered by many to be more realistic, as in many real-world multiagent settings the goal of agents is to maximize total payoffs against a variety of opponents.

We submitted bots to both competitions: Tartanian4-IRO (IRO) to the instant-runoff competition and Tartanian4-TBR (TBR) to the total bankroll competition. Both bots 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. IRO finished third in the instant-runoff competition, while TBR finished first in the total bankroll competition.

Although the bots were scored only with respect to the specific scoring rule and bots submitted to that scoring rule, all bots were actually played against each other, enabling us to compare the performances of TBR and IRO. Table 3 shows the performances of TBR and IRO against all of the bots submitted to either metric in the 2010 two-player no-limit Texas Hold'em competition.

One obvious observation is that TBR actually beat IRO when they played head-to-head (at a rate of 80 milli big blinds per hand). Furthermore, TBR performed better than IRO against every single opponent except for one (c4tw.iro). Even in the few matches that the bots lost, TBR lost at a lower rate than IRO. Thus, even though TBR uses less randomization and is perhaps more exploitable in the full game, the opponents submitted to the competition were either not trying or not able to find successful exploitations. Additionally, TBR would have still won the total bankroll competition even if IRO were also submitted.

These results show that purification can in fact yield a big gain over thresholding (with a lower threshold) even against a wide variety of realistic opponents in very large games.

8.2 Assessing worst-case exploitability in limit Texas Hold'em

Despite the performance gains we have seen from purification and thresholding, it is possible that these gains come at the expense of worst-case exploitability (see Section 4.2). Exploitabilities for several variants of a bot we submitted to the two-player limit division of the 2010 ACPC (GS6.iro) are given in Table 4; the exploitabilities were computed in the full unabstracted game using a recently developed approach [7].

Interestingly, using no rounding at all produced the most exploitable bot, while the least exploitable bot used an *intermediate* threshold of 0.15. There is a natural explanation for this seemingly surprising phenomenon. If there is too much thresholding, the resulting strategy does not have enough randomization, so it signals too much to the opponent about the agent's private information. On the other hand, if there is too little thresholding, the strategy is overfit to the particular abstraction.

Hyperborean.iro was submitted by the University of Alberta to the competition; exploitabilities of its variants are shown as well. Hyperborean's exploitabilities increased monotonically with the threshold, with no rounding producing the least exploitable bot.

Threshold	Exploitability of GS6	Exploitability of Hyperborean
None	463.591	**235.209**
0.05	326.119	243.705
0.15	**318.465**	258.53
0.25	335.048	277.841
Purified	349.873	437.242

Table 4: Worst-case exploitabilities of several strategies in two-player limit Texas Hold'em. Results are in milli big blinds per hand. Bolded values indicate the lowest exploitability achieved for each strategy.

These results show that it can be hard to predict the relationship between the amount of rounding and the worst-case exploitability, and that it may depend heavily on the abstraction and/or equilibrium-finding algorithm used. While exploitabilities for Hyperborean are more in line with what one might intuitively expect, results from GS6 show that the minimum exploitability can actually be produced by an intermediate threshold value. One possible explanation of this difference is that thresholding and purification help more when coarser abstractions (i.e., smaller abstract games relative to the full game) are used, while in finer-grained abstractions, they may not help as much, and may even hurt performance.[1] The fact that the exploitability of Hyperborean is smaller than that of GS6 suggests that it was computed using a finer-grained abstraction.

9. CONCLUSIONS

We presented two new reverse-mapping algorithms for large games: purification and thresholding. One can view these approaches as ways of achieving robustness against one's own lossy abstraction. From a theoretical perspective, we proved that it is possible for each of these algorithms to help (or hurt) arbitrarily over the standard approach, and that each can perform arbitrarily better than

[1]It is worth noting that purification and thresholding cannot help us against an equilibrium strategy if the abstraction is lossless; but even if it is lossless the algorithms may still help against actual (non-equilibrium) opponents.

	c4tw.iro	c4tw.tbr	Hyperborean.iro	Hyperborean.tbr	PokerBotSLO	SartreNL	IRO	TBR
IRO	5334 ± 109	8431 ± 156	-248 ± 49	-364 ± 42	108 ± 46	-42 ± 38		-80 ± 23
TBR	4754 ± 107	8669 ± 168	-122 ± 38	-220 ± 39	159 ± 40	13 ± 33	80 ± 23	

Table 3: Results from the 2010 Annual Computer Poker Competition for two-player no limit Texas Hold'em. Values are in milli big blinds per hand (from the row player's perspective) with 95% confidence intervals shown. IRO and TBR both use the same abstraction and equilibrium-finding algorithms. The only difference is that IRO uses thresholding with a threshold of 0.15 while TBR uses purification.

the other. However, in practice both purification and thresholding seem to consistently help over a wide variety of domains.

Our experiments on random matrix games show that, perhaps surprisingly, purification helps even when random abstractions are used. Our experiments on Leduc Hold'em show that purification leads to improvements on most abstractions, especially as the abstractions become more sophisticated. Additionally, we saw that thresholding generally helps as well, and its performance improves overall as the threshold cutoff increases, with optimal performance usually achieved at full purification. We also saw that purification outperformed thresholding with a lower threshold cutoff in the Annual Computer Poker Competition against a wide variety of realistic opponents. In particular, our bot that won the 2010 two-player no-limit Texas Hold'em bankroll competition used purification. Finally, we saw that these performance gains do not necessarily come at the expense of worst-case exploitability, and that intermediate threshold values can actually produce the lowest exploitability. There is a natural explanation for this seemingly surprising phenomenon. If there is too much thresholding, the resulting strategy does not have enough randomization, so it signals too much to the opponent about the agent's private information. On the other hand, if there is too little thresholding, the strategy is overfit to the particular abstraction.

10. FUTURE RESEARCH

Our results open up many interesting avenues for future work. In Section 6, we presented several concrete theoretical open problems related to understanding the performance of purification in random matrix games. In particular, larger games (with different degrees of abstraction) should be studied, and perhaps general theorems can be proven to augment our (statistically significant) empirical findings.

Future work should also investigate possible deeper connections between purification, abstraction, and overfitting from a learning-theoretic perspective. Is there a formal sense in which purification and thresholding help diminish the effects of overfitting strategies to a particular abstraction? Is such overfitting more prone to occur with coarser abstractions, or with some abstraction algorithms more than others? Perhaps the results also depend crucially on the equilibrium-finding algorithm used (especially for games with many equilibria). A better understanding of these phenomena could have significant practical and theoretical implications.

In addition, note that purification/thresholding is just one family of modifications to the current abstraction/equilibrium paradigm. Many other approaches are possible; for example, rounding probabilities to intermediate values (rather than to 0), or randomizing equally between the k highest-probability actions.

11. REFERENCES

[1] D. Billings, N. Burch, A. Davidson, R. Holte, J. Schaeffer, T. Schauenberg, and D. Szafron. Approximating game-theoretic optimal strategies for full-scale poker. In *IJCAI*, 2003.

[2] A. Gilpin and T. Sandholm. A competitive Texas Hold'em poker player via automated abstraction and real-time equilibrium computation. In *AAAI*, 2006.

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

[4] A. Gilpin and T. Sandholm. Expectation-based versus potential-aware automated abstraction in imperfect information games: An experimental comparison using poker. In *AAAI*, 2008. Short paper.

[5] A. Gilpin, T. Sandholm, and T. B. Sørensen. A heads-up no-limit Texas Hold'em poker player: Discretized betting models and automatically generated equilibrium-finding programs. In *AAMAS*, 2008.

[6] S. Hoda, A. Gilpin, J. Peña, and T. Sandholm. Smoothing techniques for computing Nash equilibria of sequential games. *Mathematics of Operations Research*, 35(2), 2010.

[7] M. Johanson, K. Waugh, M. Bowling, and M. Zinkevich. Accelerating best response calculation in large extensive games. In *IJCAI*, 2011.

[8] D. Koller, N. Megiddo, and B. von Stengel. Fast algorithms for finding randomized strategies in game trees. In *STOC*, 1994.

[9] R. Porter, E. Nudelman, and Y. Shoham. Simple search methods for finding a Nash equilibrium. *Games and Economic Behavior*, 63(2), 2008.

[10] D. Schnizlein, M. Bowling, and D. Szafron. Probabilistic state translation in extensive games with large action sets. In *IJCAI*, 2009.

[11] J. Shi and M. Littman. Abstraction methods for game theoretic poker. In *CG '00: Revised Papers from the International Conference on Computers and Games*. Springer-Verlag, 2002.

[12] K. Waugh, D. Schnizlein, M. Bowling, and D. Szafron. Abstraction pathologies in extensive games. In *AAMAS*, 2009.

[13] M. Zinkevich, M. Bowling, M. Johanson, and C. Piccione. Regret minimization in games with incomplete information. In *NIPS*, 2007.