

# Expectation-Based Versus Potential-Aware Automated Abstraction in Imperfect Information Games: An Experimental Comparison Using Poker

Andrew Gilpin and Tuomas Sandholm

Computer Science Department  
Carnegie Mellon University  
Pittsburgh, PA, USA

## Abstract

Automated abstraction algorithms for sequential imperfect information games have recently emerged as a key component in developing competitive game theory-based agents. The existing literature has not investigated the relative performance of different abstraction algorithms. Instead, agents whose construction has used automated abstraction have only been compared under confounding effects: different granularities of abstraction and equilibrium-finding algorithms that yield different accuracies when solving the abstracted game.

This paper provides the first systematic evaluation of abstraction algorithms. Two families of algorithms have been proposed. The distinguishing feature is the measure used to evaluate the strategic similarity between game states. One algorithm uses the probability of winning as the similarity measure. The other uses a *potential-aware* similarity measure based on probability distributions over future states. We conduct experiments on Rhode Island Hold'em poker. We compare the algorithms against each other, against optimal play, and against each agent's nemesis. We also compare them based on the resulting game's value. Interestingly, for very coarse abstractions the expectation-based algorithm is better, but for moderately coarse and fine abstractions the potential-aware approach is superior. Furthermore, agents constructed using the expectation-based approach are highly exploitable beyond what their performance against the game's optimal strategy would suggest.

## Introduction

Game-theoretic approaches to constructing agents for competitive environments have emerged as a dominant methodology in many settings, such as poker. Determining how to play game-theoretically optimally requires solving for the equilibrium of the game. Unfortunately, despite dramatic recent advances in the scalability of equilibrium-finding algorithms (Hoda, Gilpin, & Peña 2007; Gilpin *et al.* 2007; Zinkevich, Bowling, & Burch 2007; Zinkevich *et al.* 2007; McMahan & Gordon 2007; Gilpin, Peña, & Sandholm 2008), it is still impossible to scale to the size of games that are encountered in practice. For example, the game tree of heads-up (*i.e.*, two-player) limit Texas Hold'em poker has  $10^{18}$  nodes and is far beyond that scalability threshold.

Copyright © 2008, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

To handle such massive game trees, a recent practical approach has emerged: automated abstraction algorithms that take the rules of the game as input and generate a game that is much smaller and strategically similar (or, in some cases equivalent (Gilpin & Sandholm 2007b)) to the original game. For example, since 2003 there has been tremendous progress on developing computer programs for playing (both limit and no-limit) heads-up Texas Hold'em poker, and all the leading programs are nowadays developed using automated abstraction followed by equilibrium finding in the abstracted game (Zinkevich, Bowling, & Burch 2007; Gilpin, Sandholm, & Sørensen 2007; 2008).

As we will review in this paper, the automated state-space abstraction algorithms fall into two classes, *expectation-based* (Gilpin & Sandholm 2007a; Zinkevich, Bowling, & Burch 2007) and *potential-aware* (Gilpin, Sandholm, & Sørensen 2007; 2008), depending on the similarity metric used between states in the game. The existing literature has not investigated the relative performance of these two approaches. Instead, agents whose construction has used automated abstraction have only been compared under confounding effects: different granularities of abstraction and equilibrium-finding algorithms that yield different accuracies when solving the abstracted game. Therefore, it has been unclear which abstraction method is better.

In this paper, we provide the first systematic evaluation of the two classes of abstraction algorithms. Is one of them better than the other? Does the answer depend on the granularity of the abstraction that is acceptable in the output (in practice this is constrained by the scalability of the equilibrium-finding algorithm that will take the abstracted game as input)? Furthermore, does the answer depend on whether the agent competes against another agent developed using abstraction, against equilibrium play, or against its nemesis?

## Rhode Island Hold'em Poker

Poker is a game involving elements of chance, uncertainty, and strategic interaction. Optimal strategies are not straightforward, involving techniques such as misrepresentation. For these and other reasons, poker has been identified as an important challenge problem for AI (Billings *et al.* 2002).

Rhode Island Hold'em poker was invented as a testbed for computational game playing (Shi & Littman 2002). Its game tree has 3.1 billion nodes, and it shares many of the in-

interesting complications present in Texas Hold'em. Although it has been solved optimally (Gilpin & Sandholm 2007b), it remains useful in its intended role as a testbed.

Rhode Island Hold'em is a two-person zero-sum game. Thus, the equilibrium problem can be formulated as a linear program whose size is linear in the size of the game tree (Romanovskii 1962; Koller & Megiddo 1992; von Stengel 1996). Although solving these linear programs becomes difficult in practice for large games, the scalability is adequate for the abstracted games (even losslessly abstracted ones) we discuss in this paper. Hence, in our experimental evaluations, we test against the optimal strategies for the abstracted games. The tractability of finding optimal strategies under all granularities of abstraction is important because it allows us to isolate the performance of the abstraction from the performance of the equilibrium-finding algorithm (because the latter is exact here). This is the reason we use Rhode Island Hold'em—which is just under the threshold of what is solvable for equilibrium exactly—as the testbed, instead of, say, Texas Hold'em.

## Automated Abstraction

As mentioned above, two-person zero-sum sequential imperfect information games are solvable in polynomial-time in the size of the game tree. However, even with recent advances in equilibrium-finding algorithms for solving such games (Hoda, Gilpin, & Peña 2007; Gilpin *et al.* 2007; Zinkevich, Bowling, & Burch 2007; Zinkevich *et al.* 2007; McMahan & Gordon 2007; Gilpin, Peña, & Sandholm 2008), finding smaller, strategically similar representations of games remains an important component technology in the construction of game theory-based agents. The purpose of an automated abstraction algorithm is to perform this “shrinking” operation and output a much smaller but strategically similar game.

In Rhode Island Hold'em, there are 52 distinct hands in the first round,  $52 \cdot 51 = 2652$  distinct hands in the second round, and  $52 \cdot 51 \cdot 50 = 132600$  hands in the third round. The output of an automated abstraction algorithm is a mapping from these distinct hands to one of a number of buckets. The number of buckets available at each of the three rounds is parameterized by  $K_1$ ,  $K_2$ , and  $K_3$ . In our experiments we will vary the values of these parameters in order to observe the relative performances of the abstraction algorithms as we vary the granularity of the abstraction.

In our experiments, we fix the number of first-round buckets to be  $K_1 = 13$ . This value allows for an optimal bucketing at that level; it can be obtained by simply applying *suit isomorphisms*. Since each player only has one card and there are 13 different ranks, we can find the optimal 13 buckets by simply grouping the hands according to rank and ignoring the suit. Determining buckets for the second and third rounds is thus the main part of the abstraction algorithms, which we will discuss in the next two subsections. While the algorithms are not specific to any game, for convenience, we describe them in terms of how they apply to Rhode Island Hold'em.

## Expectation-based Abstraction

Currently the most sophisticated expectation-based approach to automated abstraction (Gilpin & Sandholm 2007a) uses a combination of  $k$ -means clustering (to estimate the error of different buckets) and integer programming (to allocate the buckets). When we study expectation-based abstraction, we will focus our attention on that algorithm, which we will now describe.

For the second round, the abstraction algorithm must determine how many child buckets each of the  $K_1$  first-round buckets has, subject to the constraint of having at most  $K_2$  total children. Before determining how many children each first-round bucket has, the algorithm estimates the error for each first-round bucket having various numbers of second-round children. It does this using  $k$ -means clustering, where the value of a hand is its winning probability (each tie counts as half a win).

Once these errors have been computed for all first-round buckets and for all possible values of  $k$ , the algorithm faces the allocation problem of choosing how many children each first-round bucket is allotted. (Once the number of children it can have is determined, the actual children it has will be determined by the solution to the  $k$ -means clustering problems, which the clustering algorithm has stored.) The algorithm solves this problem using a 0-1 integer program (Gilpin & Sandholm 2007a).

An analogous procedure, including the  $k$ -means clustering computations for calculating the errors and a 0-1 integer program for allocating the buckets, is repeated for the third betting round (with the  $K_2$  second-round buckets as the parents, and a limit of  $K_3$  on the maximum number of children).

## Potential-aware Abstraction

The automated abstraction algorithm presented in the previous subsection was based on a myopic expected-value (specifically, probability of winning) computation. Unfortunately, that approach ignores certain strategically relevant aspects of the game. In particular, it completely ignores the *potential* of hands. For example, in Rhode Island Hold'em poker, one high-potential hand is one in which the player has two cards of a certain suit (three are required to make a *flush*); at the present stage the hand is not very strong, but could become so if the required card showed up later in the game. *Potential-aware automated abstraction* (Gilpin, Sandholm, & Sørensen 2007) addresses this issue. The basic outline of the algorithm is the same as the expectation-based abstraction algorithm described above. The primary difference is in the metric used in comparing the strategic similarity of two games states. Instead of using winning probability as the similarity measure, the potential-aware abstraction algorithm associates with each hand a *histogram* over possible future states. The metric used to compare histograms is the  $L_2$ -distance metric. Under this approach, another design dimension of the algorithm is in the construction of the possible future states as we will discuss below.

### Potential-aware Automated Abstraction for Round 2

Recall that the algorithm is going to use  $k$ -means clustering of histograms to estimate the errors of various buckets. Before it can do the clustering, it must determine the future

possible states for the histograms. The algorithm does this for each first-round bucket by performing a *bottom-up* pass. For each first-round bucket  $i$ , it examines the *third-round* hands that contain as their first card one of the cards in first-round bucket  $i$ . Associated with each of these third-round hands is a triple  $(w, l, d)$  corresponding to the number of wins, losses, and draws a player expects when holding that hand (based on a uniform roll-out of the opponent’s card). The algorithm then performs a  $k$ -means clustering on these histograms (over win, lose, and draw counts).

The centroids that this clustering operation outputs are used as the future possible states for the second-round hands (after the second-round clustering, these temporary states are discarded). In particular, each second-round hand is associated with a histogram over these states. Next, as was done in the expectation-based algorithm above, the algorithm performs the multiple  $k$ -means clusterings for various values of  $k$  to estimate the expected error of a particular second-round bucket, given that it is allowed to have  $k$  children. Then, the second-round abstraction is computed by solving a 0-1 integer program.

### Potential-aware Automated Abstraction for Round 3

In round 3, each hand is associated with a triple  $(w, l, d)$  representing the number of wins, losses, and draws for each hand. The algorithm simply runs the multiple  $k$ -means clusterings, followed by the integer program with resource limit  $K_3$ , to determine the third round abstraction.

## Experiments

In this section we present our experimental results comparing the expectation-based and potential-aware abstraction algorithms while varying the granularity of the abstractions. We denote an abstraction with  $K_1$  first-round buckets,  $K_2$  second-round buckets, and  $K_3$  third-round buckets with the string  $K_1$ - $K_2$ - $K_3$ . For example, the abstraction granularity 13-25-125 has 13 first-round buckets, 25 second-round buckets, and 125 third-round buckets. The abstraction granularities we consider range from coarse (13-25-125) to fine (13-205-1774). At this fine granularity an equilibrium-preserving abstraction exists (Gilpin & Sandholm 2007b).

For two-person zero-sum sequential imperfect information games with perfect recall, the equilibrium problem is to find a solution to

$$\max_{x \in Q_1} \min_{y \in Q_2} x^T A y = \min_{y \in Q_2} \max_{x \in Q_1} x^T A y,$$

where  $Q_i$  is the set of *realization plans* for player  $i$  and  $A$  is the payoff matrix. The sets of realization plans are derived from the sequence form representation of the game (Koller & Megiddo 1992; Romanovskii 1962; von Stengel 1996). In the experiments we will use  $(x_*, y_*)$  to denote an optimal solution to Rhode Island Hold’em, and we will use  $(x_{EB}, y_{EB})$  and  $(x_{PA}, y_{PA})$  to denote strategies for Rhode Island Hold’em computed using the expectation-based (EB) algorithm and the potential-aware (PA) algorithm, respectively, for a given level of granularity.

Compared to the time for finding equilibria, the time needed by the abstraction algorithms was insignificant, although the potential-aware algorithm takes slightly longer than the expectation-based algorithm since the former is

doing clustering in the higher-dimensional space of histograms. We used CPLEX’s interior-point linear programming solver for finding an exact equilibrium of each of the abstracted games.

We compared the agents constructed with the two abstraction approaches using four different evaluation criteria. The following three subsections discuss three of these.<sup>1</sup>

### Comparing Agents in Head-to-head Play

The first criterion we use to compare the algorithms is to simply play the resulting strategies against each other in the full, unabstracted version of Rhode Island Hold’em. Formally, if  $(x_{EB}, y_{EB})$  is the pair of strategies computed by the expectation-based algorithm and  $(x_{PA}, y_{PA})$  is the pair of strategies computed by the potential-aware algorithm, the expected payoff to the expectation-based player is

$$\frac{1}{2} x_{EB}^T A y_{PA} - \frac{1}{2} x_{PA}^T A y_{EB}.$$

As can be seen in Table 1, the potential-aware algorithm beats the expectation-based algorithm for the abstraction granularities 13-50-250 and finer. However, interestingly, there is a cross-over: the expectation-based algorithm beats the potential-aware algorithm for the coarsest granularity 13-25-125. One hypothesis for why this is the case is that the dimensionality of the temporary states used in the bottom-up pass in the third-round (which must be smaller than the number of available second-round buckets in order for the clustering to discover meaningful centroids) is insufficient for capturing the strategically relevant aspects of the game. Another hypothesis is that since the potential-aware approach is trying to learn a more complex model (in a sense, clusters of paths of states) and the expectation-based model is trying to learn a less complex model (clusters of states, based on mere probability of winning), the former requires a larger dimension to capture this richness.

### Comparing Agents against Equilibrium Play

The second evaluation criterion is to play each of the abstractions against the optimal strategy (*i.e.*, equilibrium strategy) of the unabstracted game. If  $(x, y)$  is the pair of strategies computed by one of the abstraction algorithms and  $(x_*, y_*)$  is the optimal pair of strategies for the unabstracted game, the expected payoff to the abstracted player is

$$\frac{1}{2} x^T A y_* - \frac{1}{2} x_*^T A y.$$

The results in Table 1 show that, as expected, both algorithms improve against the optimal strategy as finer-grained abstractions are allowed. Furthermore, the potential-aware algorithm has a better payoff than the expectation-based algorithm for granularity 13-50-250 and finer, and is *optimal* for the 13-205-1774 granularity, indicating that the potential-aware algorithm finds the a *lossless* abstraction. (In fact, we verified that it finds the same abstraction as the *GameShrink* algorithm that finds lossless abstractions (Gilpin & Sandholm 2007b).) In contrast, we see that

<sup>1</sup>Our results for the fourth criterion, estimating the value of the game, are available in an extended version of this paper. Those results are basically the same as under the other three criteria.

Granularity	Experiment 1		Experiment 2		Experiment 3		
	EB Payoff	PA Payoff	EB Payoff	PA Payoff	EB Payoff	PA Payoff	PA - EB
13-25-125	16.6223	-16.6223	-25.0312	-41.7910	-160.527	-204.022	-43.495
13-50-250	-1.06272	1.06272	-19.6519	-18.2612	-134.406	-125.972	8.434
13-100-750	-6.988	6.988	-11.9801	-5.42475	-68.8882	-45.1235	23.7647
13-150-1250	-5.5703	5.5703	-6.81724	-1.57695	-59.1117	-12.067	47.0447
13-205-1774	-0.0877339	0.0877339	-0.0877339	0.0	-0.429457	-0.000460	0.428997

Table 1: Comparison of expectation-based (EB) and potential-aware (PA) abstraction against each other (Experiment 1), against the optimal strategy (Experiment 2), and against their nemeses (Experiment 3).

the expectation-based algorithm never finds a lossless abstraction, regardless of how fine an abstraction we allow it to make. This is due to the fact that sometimes two game states have exactly the same probability of winning, yet should be played differently.

### Comparing Agents against Their Nemeses: Worst-case Performance

The third criterion examines the expected worst-case performance of the algorithms. This is done by computing a *best response strategy*—i.e., a *nemesis*—for each of the two strategies, and then playing each strategy against its nemesis. If  $(x, y)$  is the strategy pair computed by one of the abstraction algorithms, the expected worst-case payoff is

$$\frac{1}{2} \min_{v \in Q_2} x^T A v + \frac{1}{2} \max_{u \in Q_1} u^T A y.$$

Table 1 shows that the performance guarantees of each of the algorithms improve as finer abstraction is allowed. Again, the potential-aware algorithm outperforms the expectation-based algorithm for abstraction granularities 13-50-250 and finer, but the expectation-based algorithm provides a better bound for the coarsest granularity.

## Conclusions and Discussion

We provided the first systematic comparison of automated abstraction algorithm for sequential imperfect information games. We examined two families of algorithms: expectation-based and potential-aware. Our experiments, conducted using the game of Rhode Island Hold'em poker (in order to isolate the abstraction issues from confounding effects), examined four criteria. The results were consistent across all four criteria. For extremely coarse abstractions, expectation-based abstraction performed the best. As the abstraction granularity becomes finer, the potential-aware algorithm outperformed the expectation-based algorithm, and was optimal in the limit. Interestingly, players generated using expectation-based abstraction are much more exploitable (by the nemesis) than the head-to-head comparisons against potential-aware abstraction would suggest.

Currently, the most successful heads-up Texas Hold'em poker-playing programs employ some form of automated abstraction followed by equilibrium computation in the abstracted game. Until now, it was not clear which method of abstraction was better. In this paper, we have shed some light on this question. Based on our experiments, it appears that for coarse abstractions, expectation-based is the best approach, and for fine-grained abstraction, potential-aware abstraction is superior. Thus, for a given game—such as Texas Hold'em—as computers become faster and

equilibrium-finding algorithms more scalable so games with finer-grained abstractions become solvable, the potential-aware approach will become the method of choice.

## Acknowledgments

This material is based upon work supported by the National Science Foundation under ITR grant IIS-0427858. We also acknowledge Intel Corporation and IBM for their gifts.

## References

- Billings, D.; Davidson, A.; Schaeffer, J.; and Szafron, D. 2002. The challenge of poker. *Artificial Intelligence* 134(1-2):201–240.
- Gilpin, A., and Sandholm, T. 2007a. Better automated abstraction techniques for imperfect information games, with application to Texas Hold'em poker. In *AAMAS'08*.
- Gilpin, A., and Sandholm, T. 2007b. Lossless abstraction of imperfect information games. *Journal of the ACM* 54(5).
- Gilpin, A.; Hoda, S.; Peña, J.; and Sandholm, T. 2007. Gradient-based algorithms for finding Nash equilibria in extensive form games. In *WINE'07*.
- Gilpin, A.; Peña, J.; and Sandholm, T. 2008. First-order algorithm with  $\mathcal{O}(\log(1/\epsilon))$  convergence for  $\epsilon$ -equilibrium in games. In *AAAI'08*.
- Gilpin, A.; Sandholm, T.; and Sørensen, T. B. 2007. Potential-aware automated abstraction of sequential games, and holistic equilibrium analysis of Texas Hold'em poker. In *AAAI'07*.
- Gilpin, A.; Sandholm, T.; and Sørensen, T. B. 2008. A heads-up no-limit Texas Hold'em poker player: Discretized betting models and automatically generated equilibrium-finding programs. In *AAMAS'08*.
- Hoda, S.; Gilpin, A.; and Peña, J. 2007. A gradient-based approach for computing Nash equilibria of large sequential games. Available at <http://www.optimization-online.org/>.
- Koller, D., and Megiddo, N. 1992. The complexity of two-person zero-sum games in extensive form. *Games and Economic Behavior* 4(4):528–552.
- McMahan, H. B., and Gordon, G. J. 2007. A fast bundle-based anytime algorithm for poker and other convex games. In *11th Int. Conf. on Artificial Intelligence and Statistics (AISTATS)*.
- Romanovskii, I. 1962. Reduction of a game with complete memory to a matrix game. *Soviet Mathematics* 3:678–681.
- Shi, J., and Littman, M. 2002. Abstraction methods for game theoretic poker. In *CG '00: Revised Papers from the Second International Conference on Computers and Games*, 333–345.
- von Stengel, B. 1996. Efficient computation of behavior strategies. *Games and Economic Behavior* 14(2):220–246.
- Zinkevich, M.; Bowling, M.; Johanson, M.; and Piccione, C. 2007. Regret minimization in games with incomplete information. In *NIPS'07*.
- Zinkevich, M.; Bowling, M.; and Burch, N. 2007. A new algorithm for generating equilibria in massive zero-sum games. In *AAAI'07*.