Tartanian7: A Champion Two-Player No-Limit Texas Hold'em Poker-Playing Program

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

Abstract

The leading approach for solving large imperfect-information games is automated abstraction followed by running an equilibrium-finding algorithm. We introduce a distributed version of the most commonly used equilibrium-finding algorithm, counterfactual regret minimization (CFR), which enables CFR to scale to dramatically larger abstractions and numbers of cores. The new algorithm begets constraints on the abstraction so as to make the pieces running on different computers disjoint. We introduce an algorithm for generating such abstractions while capitalizing on state-of-the-art abstraction ideas such as imperfect recall and the earth-mover's-distance similarity metric. Our techniques enabled an equilibrium computation of unprecedented size on a supercomputer with a high inter-blade memory latency. Prior approaches run slowly on this architecture. Our approach also leads to a significant improvement over using the prior best approach on a large shared-memory server with low memory latency. Finally, we introduce a family of post-processing techniques that outperform prior ones. We applied these techniques to generate an agent for two-player no-limit Texas Hold'em. It won the 2014 Annual Computer Poker Competition, beating each opponent with statistical significance.

1 Hierarchical Abstraction Algorithm

In order to enable distributed equilibrium finding, we have developed a new abstraction algorithm that creates an information abstraction that assigns disjoint components of the game tree to different blades so that sampling in each blade will only access information sets that are located on that blade. At a high level, the first stage of our hierarchical abstraction algorithm is to cluster public information at some early point in the game (public flop boards, i.e., combinations of public flop cards, in the case of Texas Hold'em (TH)), giving a global basis for distributing the rest of the game into non-overlapping pieces. Then, as a second stage our algorithm conducts clustering of information states (that can include both public and private information) in a way that honors the partition generated in the first stage.

The main abstraction algorithm, Algorithm 1, works as follows. Let $\hat{r}$ be the special round of the game where we perform the public clustering. For the initial $\hat{r}-1$ rounds, we compute an abstraction using an arbitrary algorithm A_r for round r. Next, the public states at round $\hat{r}$ are clustered into C buckets. Once this public abstraction has been computed, we compute abstractions for each round from $\hat{r}$ to R over all states of private information separately for each of the public buckets that have been previously computed. These abstractions can be computed using any arbitrary approach, A_r. For our poker agent, we used an abstraction algorithm that had previously been demonstrated to perform well as the A_r's (Johanson et al. 2013). To compute the abstraction of public information at round $\hat{r}$, we compute a distance between pairs of public states based on how often the states were grouped together using a strong base abstraction. We then use this distance function to compute the public abstraction using a custom clustering algorithm.

Algorithm 1 Main abstraction algorithm

```
Inputs: number of rounds $R$; round where public information abstraction is desired $\hat{r}$; number of public buckets $C$; number of desired private buckets per public bucket at round $r$, $B_r$; abstraction algorithm used for round $r$, $A_r$
  for $r = 1$ to $\hat{r} - 1$ do
      cluster information states at round $r$ using $A_r$
  cluster public information states at round $\hat{r}$ into $C$ buckets (using our custom distance function and clustering algorithm)
  for $r = \hat{r}$ to $R$ do
      for $c = 1$ to $C$ do
          cluster private information states at round $r$ that have public information in public bucket $c$ into $B_r$ buckets using abstraction algorithm $A_r$
```

2 Equilibrium-Finding Algorithm

To solve the abstract game, we developed a modification of Monte Carlo CFR (MCCFR) (Lanctot et al. 2009) specifically for architectures with high inter-blade memory access latency. It designates one blade as the "head" blade, which is used to store the regrets and average strategies for the top part of the game tree (preflop round in TH). The algorithm begins by sampling private information and conducting MCCFR on the head blade. When an action sequence is reached that transitions outside the top of the game tree (to the flop in TH), the algorithm will send the current state to each of the C child blades. Each child blade then samples public information from its public bucket, and continues the iteration of MCCFR. Once all the child blades complete their part of the iteration, their calculated values are returned to the head blade. The head blade calculates a weighted average of these values, weighing them by the number of choices of public information (possible flops in TH) that they sampled from. This ensures that the expected value is unbiased. The head node then continues its iteration of MCCFR, repeating the process whenever the sample exits the top part (a flop

SartreNLExp	Nyx	Hyperborean.iro	Slumbot	Prelude	HibiscusBiscuit	PijaiBot	Feste.iro	LittleRock	KEmpfer	Rembrant3	HITSZ_CS_14	Lucifer
261 ± 47	121 ± 38	21 ± 16	33 ± 16	20 ± 16	125 ± 44	499 ± 68	141 ± 45	214 ± 57	516 ± 61	980 ± 34	1474 ± 180	1819 ± 111

Table 1: Win rate (in mbb/h) of our agent in the 2014 AAAI Annual Computer Poker Competition against opposing agents.

sequence is encountered), until the iteration is complete.

3 New Family of Post-Processing Techniques

Post-processing techniques have been shown to be useful for mitigating the issue of overfitting the equilibrium to one's abstraction and the issue that approximate equilibrium finding may end up placing positive probability on poor actions. Two approaches have been studied, *thresholding* and *purification* (Ganzfried, Sandholm, and Waugh 2012). In thresholding, action probabilities below some threshold are set to zero and the remaining probabilities are renormalized. Purification is the special case of thresholding where the action with the highest probability is played with probability 1.

We observe that thresholding leads to the issue that discretizing actions finely in some area of the action space disfavors those actions because the probability mass from the equilibrium gets diluted among them. To mitigate this, we propose to *bucket* abstract actions into similarity classes for the purposes of thresholding. We also observe that biasing toward conservative actions that reduce variance (e.g., the fold action in poker) is helpful in a strong agent (variance increases the probability that the weaker opponent will win).

We have developed a new post-processing technique that combines these ideas. It first separates the available actions into three categories: fold, call, and bet. If the probability of folding exceeds a threshold parameter, we fold with probability 1. Otherwise, we follow purification between the three options of fold, call, and the "meta-action" of bet. If bet is selected, then we follow purification within the bet actions.

There are many variations of this technique—so it begets a family—depending on what threshold for definitely using the conservative action (fold) is used, how the actions are bucketed for thresholding, what thresholding value is used among the buckets, and what thresholding value is used within (each of possibly multiple) meta-actions.

4 Experiments

We experimented on the version of 2-player no-limit Texas Hold'em (NLTH) used in the AAAI Annual Computer Poker Competition (ACPC). We used our new abstraction algorithm to create an abstraction that is six times larger than the largest abstractions used by prior NLTH agents—and, to our knowledge, the largest imperfect-information game ever tackled by an equilibrium-finding algorithm. This scale was enabled by our new, distributed approach.

We ran our equilibrium-finding algorithm for 1,200 hours on a supercomputer (Blacklight) with a high inter-blade memory access latency using 961 cores (60 blades of 16 cores each, plus one core for the head blade), for a total of 1,153,200 core hours. Each blade had 128 GB RAM.

The results from the 2014 ACPC against all opponents are shown in Table 1. The units are milli big blinds per hand (mbb/h), and the ± indicates 95% confidence intervals. Our agent beat each opponent with statistical significance, with an average win rate of 479 mbb/h.

We compared our algorithm's performance to that of using the prior best approach on a low-latency shared-memory server with 64 cores and 512 GB RAM. This server is at the upper end of shared-memory hardware commonly available today. The approach used external sampling MCCFR to solve an abstraction computed using the state-of-the-art non-distributed algorithm (Ganzfried and Sandholm 2014).

We benchmarked both against the two strongest agents from the 2013 competition, Figure 1. The new approach outperformed the old against both agents for all timestamps tested. So, it is able to effectively take advantage of the additional distributed cores and RAM.

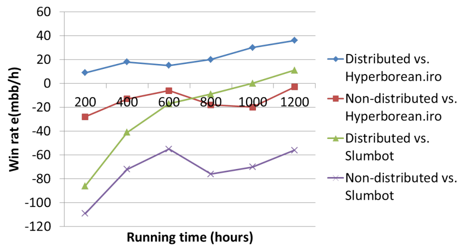

Figure 1: Trajectory of win rates over time against the two strongest agents from the 2013 poker competition.

We also studied the effect of using our new post-processing techniques on the final strategies computed by our distributed equilibrium computation. We compared using no threshold, purification, a threshold of 0.15, and using the new technique with a threshold of 0.2. We tested against the same two strongest agents from the 2013 competition. Results are shown in Table 2. The new post-processor outperformed the prior ones both on average performance and on worst observed performance.

	Hyperborean.iro	Slumbot	Avg	Min
No Threshold	+30 ± 32	+10 ± 27	+20	+10
Purification	+55 ± 27	+19 ± 22	+37	+19
Thresholding-0.15	+35 ± 30	+19 ± 25	+27	+19
New-0.2	+39 ± 26	+103 ± 21	+71	+39

Table 2: Win rate (in mbb/h) of several post-processing techniques against the strongest 2013 poker competition agents.

References

Ganzfried, S., and Sandholm, T. 2014. Potential-aware imperfect-recall abstraction with earth mover's distance in imperfect-information games. In *AAAI*.

Ganzfried, S.; Sandholm, T.; and Waugh, K. 2012. Strategy purification and thresholding: Effective non-equilibrium approaches for playing large games. In *AAMAS*.

Johanson, M.; Burch, N.; Valenzano, R.; and Bowling, M. 2013. Evaluating state-space abstractions in extensive-form games. In *AAMAS*.

Lanctot, M.; Waugh, K.; Zinkevich, M.; and Bowling, M. 2009. Monte Carlo sampling for regret minimization in extensive games. In *NIPS*.