Potential-Aware Imperfect-Recall Abstraction with Earth Mover's Distance in Imperfect-Information Games*

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

Abstract

There is often a large disparity between the size of a game we wish to solve and the size of the largest instances solvable by the best algorithms; for example, a popular variant of poker has about 10^{165} nodes in its game tree, while the currently best approximate equilibrium-finding algorithms scale to games with around 10^{12} nodes. In order to approximate equilibrium strategies in these games, the leading approach is to create a sufficiently small strategic approximation of the full game, called an abstraction, and to solve that smaller game instead. The leading abstraction algorithm for imperfect-information games generates abstractions that have imperfect recall and are distribution aware, using k-means with the earth mover's distance metric to cluster similar states together. A distribution-aware abstraction groups states together at a given round if their full distributions over future strength are similar (as opposed to, for example, just the expectation of their strength). The leading algorithm considers distributions over future strength at the final round of the game. However, one might benefit by considering the trajectory of distributions over strength in all future rounds, not just the final round. An abstraction algorithm that takes all future rounds into account is called potential aware. We present the first algorithm for computing potential-aware imperfect-recall abstractions using earth mover's distance. Experiments on no-limit Texas Hold'em show that our algorithm improves performance over the previously best approach.

1 Introduction and Background

There is often a large disparity between the size of a game we wish to solve and the size of the largest instances solvable by the best algorithms. For example, two-player no-limit Texas Hold'em poker with common stack sizes (used in the Annual Computer Poker Competition) has about 10^{165} nodes in its game tree (Johanson 2013), while the currently best approximate equilibrium-finding algorithms scale to games with around 10^{12} nodes (Zinkevich et al. 2007; Hoda et al. 2010). In order to approximate equilibrium strategies in these games, the leading approach is to create a sufficiently small strategic approximation of the full game, called an *abstraction*, and to solve that smaller game instead. Abstraction in games is quite different than in single-agent settings. For example, it is not monotonic: if we refine an abstraction, we may get strategies that have higher exploitability in the original game (Waugh et al. 2009a). Abstraction has been successfully applied to two-player Texas Hold'em poker, first with a manually generated abstractions (Shi and Littman 2002; Billings et al. 2003), and now with abstraction algorithms (Gilpin and Sandholm 2006; 2007a; Gilpin, Sandholm, and Sørensen 2008; Waugh et al. 2009b; Johanson et al. 2013). Many abstraction algorithms work by coarsening the moves of chance, merging several information sets of the original game into single information sets of the abstracted game. Additional related work on game abstraction is discussed in Section 5.

One approach for determining which information sets should be grouped together is to come up with a measure of 'strength' for each state, then run a clustering algorithm, such as k-means, using the difference in 'strength' as the distance metric. For example, in poker, a natural measure of a hand's strength is the *equity* (probability of winning plus one-half the probability of tying) against a uniform random draw of private cards for the opponent, assuming a uniform random rollout of the remaining public (i.e., shared) cards. This is also known as the *expected hand strength (EHS)* metric. For example, if a player is dealt two aces as his two private cards in Texas Hold'em, he will win 84.93% of the time and will tie 0.55% of the time against a random hand assuming a random rollout of the public cards, giving his hand an equity of 0.852. Similarly, the equity of two kings is 0.824. Since these two values are similar, it is likely that they would be grouped together by a clustering algorithm. Early approaches for abstraction in poker used EHS (or EHS exponentiated to some power) to cluster hands (Billings et al. 2003; Gilpin and Sandholm 2006; Waugh et al. 2009b; Zinkevich et al. 2007).

While EHS is a reasonable first-order-approximation for the strength of a hand, it fails to account for the entire probability distribution of hand strength. For example, the hands KcQc (king and queen of clubs) and 6c6d (six of clubs and six of diamonds) have expected hand strengths of 0.634 and 0.633 respectively, which suggests that they have very similar strength. However, looking at the full distributions of

*Please cite the conference version: "Ganzfried, S., and Sandholm, T. 2014. Potential-Aware Imperfect-Recall Abstraction with Earth Mover's Distance in Imperfect-Information Games. In Proceedings of the *AAAI Conference on Artificial Intelligence (AAAI)*."

hand strength, as opposed to just its expectation, paints a very different picture, as previous work has shown (Gilpin, Sandholm, and Sørensen 2007; Johanson et al. 2013). Figures 1 and 2 (which are similar to part of Figure 2 from Johanson et al.'s work) show the full histograms of expected hand strength for the two hands, where each bar corresponds to the probability mass of the given level of hand strength. For example, if the public board cards are 7dQh4h2s3c, then KcQc has an equity of 0.856 against a uniform random opponent hand; so the histogram entry corresponding to the column for an equity of 0.84–0.86 is incremented by one for this hand (prior work has assumed that the equities are divided into 50 regions of size 0.02, as do these figures). As the figures indicate, despite the fact that these hands have similar expected hand strengths, their full distributions are very different. For example, 6c6d frequently has an equity between 0.5 and 0.7 and rarely has an equity between 0.7 and 0.9, while the reverse is true for KcQc.

An abstraction algorithm that considers the full distributions of hand strength, as opposed to just the expectation, is called *distribution aware*. The leading abstraction algorithm for imperfect-information games generates abstractions that are distribution aware (Johanson et al. 2013), and it has been shown empirically that the distribution-aware approach significantly outperforms EHS-based approaches (Gilpin and Sandholm 2008; Johanson et al. 2013). The natural metric for computing distances between histograms of hand-strength distributions is the *earth mover's distance (EMD)*. 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." Earlier work on distribution-aware abstraction used the L_2 distance metric instead (Gilpin, Sandholm, and Sørensen 2007), which has been shown to be significantly less effective because it does not properly account for how far the "dirt" needs to be moved (only how much needs to be moved). Using one-dimensional histograms as done above, EMD can be computed by a straightforward linear time procedure that scans the histogram and keeps track of how much dirt needs to be transported between consecutive bins. However, computing EMD is much more challenging as the dimensionality of the data increases, and as we will present later, multi-dimensional EMD computation will be needed in more sophisticated abstraction algorithms.

In the domain of Texas Hold'em poker,[1] the leading abstraction algorithm works as follows (Johanson et al. 2013). In the first round, there is no card abstraction, and each hand is in its own bucket. In the second and third rounds, abstractions are computed as follows. First, an equity histogram is constructed for each hand, similarly to those in Figures 1 and 2. For example, for the flop, we will create a histogram for the hand where the private cards are Kc3h and the public cards are KsTd8h. Then k-means is used to compute an abstraction with a desired number of clusters, using the EMD between each pair of histograms as the distance metric. One important feature of these abstractions is that they have *imperfect recall*: a player can be made to forget information that he knew earlier in the hand. For example, the hands Kc3h-KsTd8h and Kc4h-KsTd8h will likely be grouped together on the flop, even though the player could distinguish between Kc3h and Kc4h in the preflop round. Imperfect recall abstractions have been demonstrated to lead to significantly stronger performance than perfect recall for an abstraction of a given size, because they allow the player to have a more refined view of the present since he is allowed to forget details about the past (Waugh et al. 2009b).[2] That algorithm computes abstractions for the flop and turn rounds independently using this approach. It computes the abstraction for the final round using a different approach (k-means with L_2 over vectors of EHS against first-round clusters of the opponent).

As described above, the equity histograms for the flop (and turn) rounds consider distributions over future strength at the final round of the game (i.e., after all the public cards are dealt). However, it is possible that two flop hands have very similar distributions over strength after the river is dealt, but they realize the equity at very different rates throughout the hand. Section 2 provides example situations of how this can arise, both in poker and in a general domain-independent game. Thus, a natural direction to explore is whether one might benefit by considering the distribution over strength in all future rounds, not just the final round. An abstraction algorithm that takes all future rounds into account is called *potential aware*. Prior work on potential-aware abstraction (Gilpin, Sandholm, and Sørensen 2007) applied only to perfect-recall abstraction and used the L_2 distance metric, both of which have significant shortcomings, as described above.

In this paper, we present the first algorithm for computing potential-aware imperfect-recall abstractions, using EMD as the distance metric. We design a new abstraction algorithm that combines these three threads, each of which has been shown helpful separately in the past. Computing imperfect-recall abstractions is significantly more challenging than in the perfect-recall case, since the set of hands that must be clustered at each step is much larger. Additionally, computing EMD in this setting is significantly more challenging than in the one-dimensional distribution-aware setting, and is also much more challenging than computing L_2 distance in the potential-aware setting. The best commercially-available algorithm for computing (multi-dimensional) EMD (Pele and Werman 2008; 2009) is far too slow to compute abstractions in poker, and we develop a fast custom heuristic for approximating EMD in our setting. Experiments on no-limit Texas Hold'em show that our algorithm leads to a statistically significant improve-

[1] Texas Hold'em contains four betting rounds. In the first round (*preflop*), players are each dealt two private cards; in the second round (the *flop*), three public cards are dealt on the table; in the third round (the *turn*), one more public card is dealt; and in the final round (the *river*), one final public card is dealt. If play makes it to the end of the final round, then the player with the best five-card hand (out of his two private cards and the five public cards) wins the pot.

[2] A downside of using imperfect-recall abstractions is that they typically cause equilibrium-finding algorithms to lose their convergence guarantees.

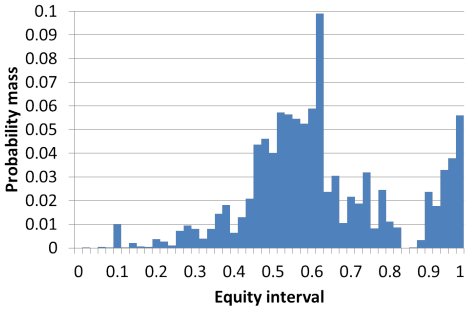

Figure 1: Equity distribution for 6c6d.

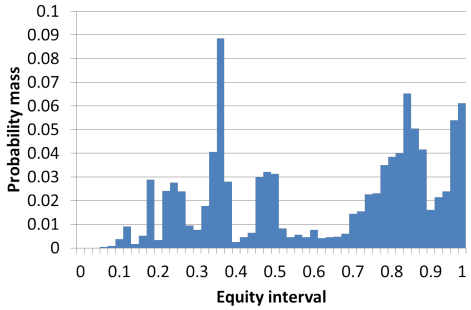

Figure 2: Equity distribution for KcQc.

ment in performance over the previously best abstraction algorithm.

2 Potential-Aware Abstraction

In this section, we present examples that demonstrate the difference between potential-aware abstraction and the leading distribution-aware approach, which considers distributions over future strength at the *final* round of the game. The examples show that it is possible for two different states of private information to have very similar (even identical) histograms over equity at the end of the game, but to realize this equity in very different ways throughout the play of the game. We first present a domain-independent example in Section 2.1, followed by an example of a poker situation demonstrating this phenomenon in Section 2.2.

2.1 Domain-Independent Example

We consider the following game. A player is initially given private signal x_i, and then chance makes up to two moves before the game ends. The information equity trees for x_1 and x_2 are given in Figures 3 and 4. The opponent is also given a private signal from some distribution, and the equities of the initial player against the opponent's distribution are given in the leaves of the trees. If the player has x_1, then chance selects the right branch with probability 1 in the first round, then selects each branch with probability $\frac{1}{2}$ in the second round. If chance selects the right branch in the second round, then the player has an equity of 1; otherwise, he has equity 0. If the player has x_2, then chance selects each branch with probability $\frac{1}{2}$ in the first round, and selects the right branch with probability 1 in the second round (for each choice of actions in the first round). If chance selected the left branch in the first round, then the player's equity is 0; if chance selected the right branch in the first round, then his equity is 1.

If we use the traditional distribution-aware approach of considering equity assuming the end of the game is reached, then equity histograms for both x_1 and x_2 are the same and given in Figure 5: with probability $\frac{1}{2}$, the player will have equity 0, and with probability $\frac{1}{2}$, he will have equity 1. For this example, we assume that the equities are broken into five equally-sized intervals. (Several of the strongest poker agents use fifty intervals each of width 0.02.) Since these histograms are identical, the EMD between the two states corresponding to the two private signals respectively would be zero, so they would be treated as being identical.

However, these two states are actually quite different if we consider how the equity changes between the first and second round, as opposed to just jumping to the end of the game. With x_2, the player will know for sure whether he has an equity of 0 or 1 after chance's first move, while with x_1 he will not. To perform potential-aware abstraction, the first step is to compute the histograms for both players at the possible states in the second round. The histogram for x_1 after chance selects the right branch (i.e., at B) is also the histogram given in Figure 5; the histogram for x_2 after chance selects the left branch (i.e., at D) has unit mass in the left-most column (equity of 0–0.2); and the histogram for x_2 after chance selects the right branch (i.e., at E) has unit mass in the right-most column (equity of 0.8–1).

Next, we compute the histograms for x_1 and x_2 at the first round, with respect to the possible states that could be reached at the second round. The histogram for x_2 is given in Figure 6. The possible states B, D, and E correspond to the second round states in the information equity trees. We omit additional states that could have originated from x_3, x_4, etc. (they will all have probability 0). As the figure shows, with x_2 the player will be in states D and E with probability $\frac{1}{2}$. The histogram for x_1 will have unit mass in the column for state B. Unlike the histograms above, whose x-axis is cardinal (i.e., equity), the x-axis of these histograms is not even ordinal (the next-round states can be listed in any arbitrary order).

To transform this new histogram for x_1 into the histogram for x_2, we must move a mass of $\frac{1}{2}$ from the B column to both the D and E columns. Thus, the EMD is $\frac{1}{2}d(B,D)+\frac{1}{2}d(B,E)$, where the *ground distances* $d(B,D)$ and $d(B,E)$ are computed using the second-round histograms described above. To transform the histogram at B into the histogram at D, we must move a mass of $\frac{1}{2}$ from the rightmost column to the leftmost column; so $d(B,D) = \frac{1}{2} \cdot 4 = 2$. Similarly, $d(B,E)$ also equals 2. So the EMD between the two, non-ordinal first-round histograms is $\frac{1}{2} \cdot 2 + \frac{1}{2} \cdot 2 = 2$. Thus, potential-aware abstraction will treat x_1 and x_2 differently, and potentially group them into different clusters (while the distribution-aware approach will treat

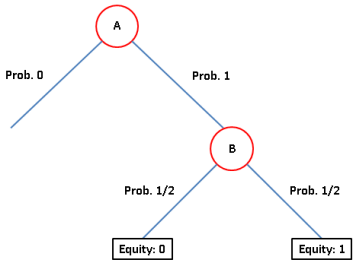

Figure 3: Information equity tree for private signal x_1.

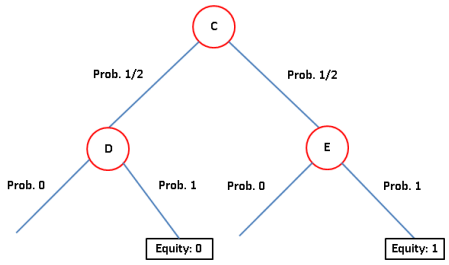

Figure 4: Information equity tree for private signal x_2.

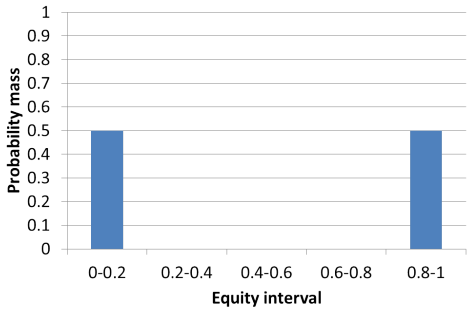

Figure 5: Histogram of equity for both private information x_1 and x_2 at round 1, assuming the game reaches the end of the final round.

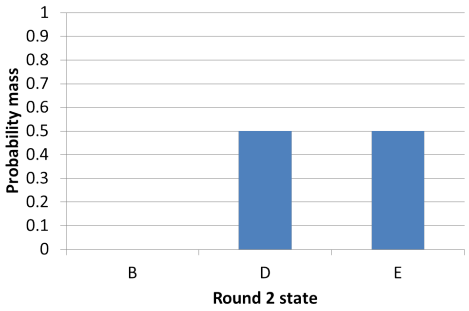

Figure 6: Histogram for private signal x_2 at round 1 over non-ordinal information states at round 2.

them as identical, as shown above).

2.2 Poker Example

In Section 1 we provided a canonical example of two Texas Hold'em hands with similar EHS, but very different histograms over equity at the end of the hand (KcQc vs. 6c6d). We now present an example of two hands that have similar histograms over equity at the final round (and thus also similar EHS), but realize their equity in very different ways throughout the hand.

Consider the two flop hands TcQd-7h9hQh and 5c9d-3d5d7d (the first two cards are the private cards, and the next three are the public flop cards). These are both relatively strong hands. The first hand has top pair (a pair of queens), and the second hand has second pair (a pair of fives) plus a flush draw (another diamond on the board would complete a flush). The hands both have very similar EHS, assuming both the turn and river are dealt; TcQd-7h9hQh has EHS 0.683 and 5c9d-3d5d7d has EHS 0.679. These two hands also have very similar full distributions over equity at the final round, as can be seen in Figures 7 and 8. The EMD between these two distributions is 0.559, where the unit is equity intervals (assuming fifty intervals of width 0.02, and a total unit mass for both distributions). This value is sufficiently low that these two hands are grouped into the same bucket by the leading distribution-aware abstraction algorithm (and therefore, they would be treated as being identical by equilibrium-finding algorithms).

However, despite similarities between the equity distributions after the river is dealt, these two hands realize their equity very differently throughout the hand. Figures 9 and 10 show their expected hand strength distributions on the turn (the next public card), assuming a uniform random distribution for the river card and the opponent's cards. These two distributions are very different; for example, a large portion of the time TcQd-7h9hQh will have a turn equity between 0.68 and 0.78, while 5c9d-3d5d7d has a large portion of its equity in the 0.56–0.66 range.

We note that Figures 9 and 10 depict the distributions of expected turn hand strength, as opposed to full distributions over distributions of river hand strength (since each turn card will lead to a distribution over strength in the next round, not a single value). For example, for the hand TcQd-7h9hQh, if the turn card is Ad, the hand's expected equity, assuming a uniform random river card and uniform random hand for the opponent, is 0.695 (though it will vary for individual river cards); so the interval for 0.68–0.7 in the histogram would be incremented by one for that turn card. This is significantly more simplistic than our potential-aware approach, which takes into account the full distribution of turn 'buckets', which are themselves distributions over equity intervals after the river. However, comparing these distributions is still useful for several reasons. First, if the full distributions over turn hand strength (which are, themselves, distributions over river hand strength) were similar, then the distributions over the expectation of turn hand strength distributions would necessarily be similar as well; thus, the fact that the expectation distributions differ significantly indicates that the full distributions also differ significantly. And second, it is not feasible to compactly represent the full distributions visu-

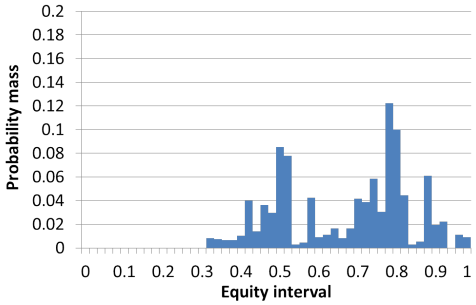

Figure 7: Equity distribution for TcQd-7h9hQh on the river (final betting round).

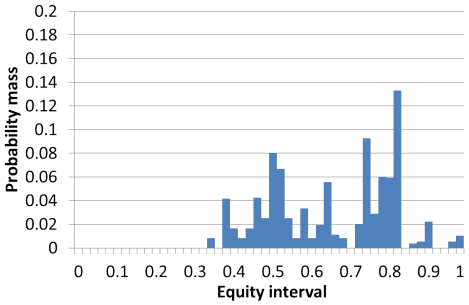

Figure 8: Equity distribution for 5c9d-3d5d7d on the river (final betting round).

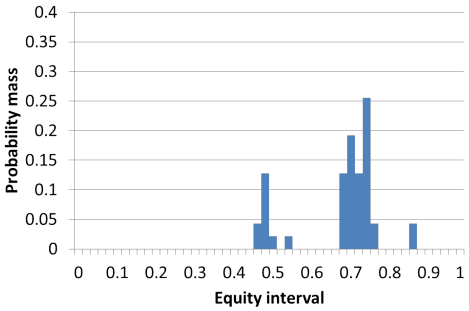

Figure 9: Equity distribution for TcQd-7h9hQh on the turn (next betting round). Each point is the expected hand strength for a given turn card assuming a uniform random distribution for the river card and the opponent's cards.

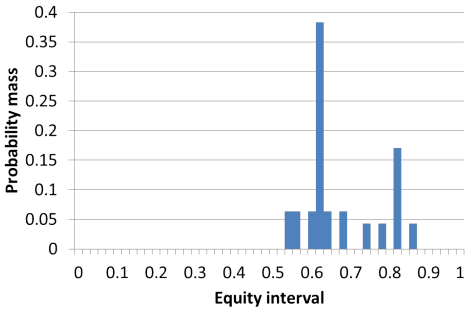

Figure 10: Equity distribution for 5c9d-3d5d7d on the turn (next betting round). Each point is the expected hand strength for a given turn card assuming a uniform random distribution for the river card and the opponent's cards.

ally, while the distributions of expected hand strength can be represented easily as two-dimensional histograms.

While the EMD between the river equity distributions depicted in Figures 7 and 8 is 0.559, the EMD between full turn distributions using the potential-aware approach is 4.519 (using comparable units). Potential-aware abstraction is able to correctly identify that these hands are quite different, and places them into different buckets due to their large EMD (while the prior abstraction algorithm places them in the same bucket).

3 Algorithm for Potential-Aware Imperfect-Recall Abstraction, with EMD

In this section, we present our new algorithm for computing potential-aware imperfect-recall abstractions using EMD. We first present our main algorithm, followed by a heuristic for quickly approximating the EMD in our setting that we use to make the algorithm practical for large games such as Texas Hold'em.

Our abstraction algorithm, depicted in Algorithm 1, works as follows. Assume the information tree has $r+1$ levels (0–r), and for each level n, a number of clusters C^n is specified as input. For the final rounds $n = \hat{r}, \ldots, r$, an arbitrary abstraction algorithm S^n is used, with distance function d^n, to produce C^n clusters; let A^n denote the resulting abstraction, and let m_i^n denote the mean of the i'th cluster in A^n. Next, we compute the abstraction at round $\hat{r}-1$ as follows. First, we compute the distance $d_{i,j}^n$ between each pair of round-(n+1) means m_i^{n+1} and m_j^{n+1}, using the distance metric d^{n+1}. Next, we compute histograms $H^n(x^n)$, where the i-th element of $H^n(x^n)$ is the fraction of the time that chance's next move will send x^n into cluster i in A^{n+1}. Finally, we compute the abstraction A^n at round n, by clustering the histograms H^n into C^n clusters using clustering algorithm L^n (prior work in poker uses k-means). The distance metric used, denoted d^n, is the EMD between the histograms, using $d_{i,j}^n$ as the ground distance between components i and j of a histogram. We then compute the new cluster means m_i^n, and continue in the same fashion for $n = \hat{r}-2, \ldots, 0$. The resulting abstraction, A^n, has imperfect recall since we cluster all of the histograms $H^n(x^n)$ without any regard for the information known at the corresponding states at previous stages of the game, and potentially we cluster two states together that contain information that we could distinguish between at earlier rounds.

To compute the distances in the main loop of Algorithm 1 we implemented the fastest commercially-available multi-dimensional EMD algorithm (Pele and Werman 2008; 2009); however, it was far too slow for the domain of Texas Hold'em. So we were forced to develop a faster heuristic for

Algorithm 1 Main algorithm for computing potential-aware imperfect-recall abstractions

Inputs: $\{C^n\} : n = 0, \ldots, r$; $\{S^n\}, \{d^n\} : n = \hat{r}, \ldots, r$; $\{L^n\} : n = 0, \ldots, \hat{r} - 1$

for $n = r$ **to** $\hat{r}$ **do**
  Compute abstraction A^n with C^n clusters using abstraction algorithm S^n with distance function d^n
end for
for $n = \hat{r} - 1$ **to** 0 **do**
  for $i = 1$ **to** C^n **do**
    $m_i^{n+1} \leftarrow$ mean of cluster i in A^{n+1}
  end for
  for $i = 1$ **to** $C^n - 1$ **do**
    for $j = i + 1$ **to** C^n **do**
      $d_{i,j}^n \leftarrow$ distance between m_i^{n+1} and m_j^{n+1} using distance function d^{n+1}
    end for
  end for
  for each point x^n at round n **do**
    $H^n(x^n) \leftarrow$ histogram for x^n over clusters from A^{n+1}
  end for
  Compute abstraction A^n over histograms H^n using clustering algorithm L^n and distance function d^n (i.e., EMD with $d_{i,j}^n$ as the ground distance) to produce C^n clusters
end for

approximating EMD in this setting. Our heuristic is given in Algorithm 2. The context is that we are running k-means to compute an abstraction at level n of the tree, for which we must compute the distance between each 'point' and each mean. The 'points' correspond to the histograms $H^n(x^n)$ over clusters in A^{n+1}, and were computed in the previous step of Algorithm 1. The naïve way of representing them would be as vectors of dimension C^{n+1}. However, this vector may be very sparse. For example, if $C^{n+1} = 5000$ (as in our experiments), but the current point can only transition into 50 next-round clusters with positive probability, we would like to take advantage of a sparser representation rather than represent it as a vector of size 5000. Instead, we represent the point as a vector of length 50 (in this example), where each index corresponds to the index of the next-round cluster we transition to. For example, if a point can transition to clusters 3, 5, or 10, for different chance moves, then we represent the point as the vector $(3, 5, 10)$, where each of these will have probability $\frac{1}{3}$. Each point x^n in Algorithm 2 corresponds to such a vector, where N denotes the length. For simplicity we assume that all of the elements are distinct, though repeated elements can be dealt with straightforwardly.

We similarly take advantage of sparsity in representing the means. While each mean could potentially have C^{n+1} entries, many of these entries may be zero. Instead, we simply represent the mean m as the vector of the nonzero entries, of which we assume there are Q. In order to identify which clusters the entries correspond to, and to make

Algorithm 2 Algorithm for efficiently approximating EMD in our setting

Inputs: Point x^n with N elements; mean m with Q elements; sortedDistances[i][j], orderedClusters[i][j], for $1 \leq i \leq C^{n+1}, 1 \leq j \leq Q$

targets[] $\leftarrow$ array of size N with all elements equal to $\frac{1}{N}$
meanRemaining[] $\leftarrow$ copy of m
done[] $\leftarrow$ array of size N with all elements set to false
totCost $\leftarrow 0$
for $i = 1$ **to** Q **do**
  for $j = 1$ **to** N **do**
    if done[j] == **true then**
      continue
    end if
    pointCluster $\leftarrow x^n[j]$
    meanCluster $\leftarrow$ orderedClusters[pointCluster][i]
    amtRemaining $\leftarrow$ meanRemaining[meanCluster]
    if amtRemaining == 0 **then**
      continue
    end if
    $d \leftarrow$ sortedDistances[pointCluster][i]
    if amtRemaining $<$ targets[j] **then**
      totCost += amtRemaining * d
      targets[j] -= amtRemaining
      meanRemaining[meanCluster] $\leftarrow 0$
    else
      totCost += targets[j] * d
      targets[j] $\leftarrow 0$
      meanRemaining[meanCluster] -= targets[j]
      done[j] $\leftarrow$ true
    end if
  end for
end for
return totCost

our overall implementation more efficient, we utilize several data structures. First, we precompute an array called sortedDistances, where sortedDistances[i][j] is the distance between next-round cluster i and the j-th closest cluster to i for which the current mean has non-zero probability, where distances have already been computed using d^{n+1}. We also use a structure orderedClusters, where orderedClusters[i][j] is the index of the cluster that the mean assigns non-zero probability to that is j-th closest to cluster i. These arrays are precomputed in advance of the EMD computation; while they require some time to compute, the EMD computations are by far the bottleneck of the algorithm. This additional computation helps us overall since it significantly speeds up the EMD computations.

Given these data structures as input, we now approximate EMD as follows. First, we start with the first entry of x^n; we call the round-(n+1) cluster to which this belongs 'pointCluster.' We then find the closest cluster to pointCluster that corresponds to a nonzero element in the mean; this will be the element orderedClusters[pointCluster][1], which we call 'meanCluster.' We shift as much mass as possible between from this mean element to the corresponding point element.

The cost is increased by the amount of mass we shift multiplied by the distance. We then update the remaining point mass and mean mass at the elements we have considered, and continue for the remaining point indices $j = 2, \ldots, N$. Next, we set $i = 2$, and repeat the same procedure, now shifting as much mass as is available from the second closest nonzero mean cluster to each cluster of the point. We repeat this for $i = 3, \ldots, Q$, until the mean vector has been fully transformed into the point vector. We then output the resulting total cost.

As mentioned above, the fastest commercially-available algorithm for computing EMD (Pele and Werman 2008; 2009) is far too slow to be effective in Texas Hold'em. (This was despite the fact that we integrated the data structures described above with this algorithm to exploit sparsity, as well as applied several other enhancements to improve performance of k-means, such as a pruning technique that exploits the triangle inequality (Elkan 2012) and parallelizing each step using 64 cores.) For the point-mean distances in the first round of k-means, the exact EMD algorithm averaged 11.4 ms per computation, while our heuristic averaged only 0.008 ms per computation. Furthermore, the exact algorithm scales extremely poorly as the dimensionality of the inputs increases. Since the initial means are themselves data points, their dimensionality is small; however, for future rounds of k-means, the means are weighted averages over all points in their cluster, and have higher dimensionality. Our new algorithm performs well even in the future rounds of k-means as this dimensionality increases, while the exact algorithm scales very poorly.

There are many potential further improvements to approximating EMD and doing clustering in this context. However, even with the techniques we already developed and tested, the approach outperforms the previously best abstraction algorithm, as the experiments in the next section will show.

4 Experiments

We evaluated our abstraction algorithm in a very large sequential imperfect-information game, two-player no-limit Texas Hold'em. While our abstraction algorithm works for any number of agents and does not assume a zero-sum game, we focused on this two-player zero-sum game in the experiments—as is most common in this field—so that we can compute a near equilibrium to a large abstraction, and thereby evaluate the results.

4.1 Head-to-Head Performance vs. Best Prior Abstraction Algorithm

We ran two different sets of experiments corresponding to two different manually-generated betting abstractions. In both experiments, we compared performance to the previously best abstraction algorithm (Johanson et al. 2013). In each experiment, we used the same betting abstraction for us and for the benchmark. In the first experiment, we used the betting abstraction that was used by the agent that finished in 2nd place in the 2012 Annual Computer Poker Competition. We chose to test on this relatively small betting abstraction so that we could obtain a good level of convergence to equilibrium in the abstract game. In the second experiment, we used a very large betting abstraction; it is about 8 times larger than the version used by our 2013 competition agent, and currently beats the winner from the 2013 competition.

In both experiments, we used 169, 5000, 5000, and 5000 card buckets respectively in the four betting rounds for both the new algorithm and the prior algorithm. This card abstraction is used by our strongest agent that now beats the winner from the 2013 competition, and was also used by the 2013 competition agent that finished in 3rd. Also, as is typical nowadays among all the top teams, the first round has 169 buckets corresponding to no abstraction at all.

In each of the two experiments, we created an agent that was identical to the corresponding opponent, except that it used our new algorithm to compute the abstraction for the flop round (i.e., second betting round). For both flop abstraction algorithms, we conducted 25 restarts using the k-means++ initialization procedure (Arthur and Vassilvitskii 2007), and selected the run that produced the lowest within-cluster sum of squares. We chose to focus on the flop round for the following reasons. First, the strongest agents do not use any abstraction preflop (i.e., on the first betting round), and there is no potential on the river (i.e., last betting round) since no further cards will be dealt; so potential-aware abstraction would not help on those rounds. The approach is potentially useful on the turn (i.e., third betting round) as well, but it appears to be computationally intractable, even using our fast heuristic for EMD (there are around 1.3 million hands to be clustered on the flop, and 55 million on the turn). For each of the generated abstractions (two new and two benchmarks), we computed an approximate equilibrium for the abstraction using a sampled version of counterfactual regret minimization (Lanctot et al. 2009).

In each of the two experiments, we ran 20,000 duplicate matches between our new agent and the respective benchmark agent. In both experiments, our new approach led to a statistically significant improvement over the old approach. In the first experiment, the new agent beat its benchmark by 2.58 milli big blinds per hand (mbb/h) with a 95% confidence interval of ± 1.56 mbb/h. In the second experiment, the new agent beat its benchmark by 2.22 mbb/h (± 1.28 mbb/h).

4.2 Evaluating the Approximation of Potential-Aware EMD

To evaluate how closely the distance computed by Algorithm 2 approximates the true potential-aware EMD, we repeatedly generated the histograms (over turn buckets) for two random flop hands, and computed the exact EMD between the histograms. If this distance was less than some threshold, then we also ran Algorithm 2 to approximate the EMD. (We chose a threshold of 3000[3] since it appears that the vast majority of the closest point-mean distances were in the 0–2500 range, and we are mostly interested in how well our heuristic does at approximating EMD for the point-mean pairs with low distance, since those represent the clus-

[3]These values must be divided by 1081 for the total histogram mass to be normalized to one.

ter to which a point might actually be assigned. We are not as concerned about how far off our heuristic is for distances that are extremely large, as they will likely be pruned and have no chance of being assigned as the closest mean.) We computed the relative error between the EMD computed by our heuristic and the true EMD, and averaged it over many samples. Our algorithm had average relative error of 0.1496 (with 95% confidence interval ± 0.0014).

For comparison, we also computed the EMD between the same flop hands using the previously best distribution-aware approach, where the histograms consider equity assuming the end of the game is reached. That approach produced an average relative error of 0.2084 (± 0.0014) compared to the potential-aware EMD over the same sample. Thus, our potential-aware, but heuristically-computed EMD, approximates the true potential-aware EMD 28.2% better than the prior approach for computing EMD, which used exact calculation but was not potential aware.

Though we already demonstrated the superiority of our abstraction algorithm over the prior algorithm in the experiments described in Section 4.1, these results provide a further sanity check that Algorithm 2 does in fact lead to a better degree of approximation of potential-aware EMD than the prior method. The results also indicate that there is still room for significant improvement toward more accurate potential-aware EMD computation.

5 Additional Related Research

The closely related research on game abstraction was already cited earlier. Here we discuss additional papers on game abstraction.

The GameShrink algorithm computes a lossless information abstraction (Gilpin and Sandholm 2007b) in a class of games, but in very large games like Texas Hold'em, lossy abstraction is needed to create an abstract game that is small enough to solve to near equilibrium.

This paper, like most prior work on game abstraction, has been about information abstraction. Typically action abstraction (e.g., selection of bet sizes to include in the abstraction in poker) is done manually—as was done in this paper as well. There is also an emerging thread of research into automated action abstraction (Hawkin, Holte, and Szafron 2011; 2012; Brown and Sandholm 2014).

Recently, a framework and algorithms were presented for lossy abstraction with bounds on the resulting equilibrium quality. It applies both to information and action abstraction. This is for stochastic games (Sandholm and Singh 2012), and more recently extensive-form games (Kroer and Sandholm 2014). For extensive-form games, the algorithm—an integer program—only scales to a tiny poker game (with a five-card deck and two rounds of betting with a fixed bet size), so the algorithm is not applicable to large poker variants, such as Texas Hold'em. That framework does apply to Texas Hold'em, and it remains an open question whether scalable algorithms within that framework can be developed. Other recent work provides a regret bound in the full game for a class of games for strategies produced by applying an equilibrium-finding algorithm (counterfactual regret minimization) to abstractions (Lanctot et al. 2012). The regret bound applies for both perfect and imperfect recall abstractions of games from the given class.

6 Conclusions and Future Research

We presented the first algorithm for computing potential-aware imperfect-recall abstractions using earth mover's distance as a distance metric. This is the first algorithm that combines potential-aware abstraction with imperfect recall. It is also the first time earth mover's distance has been used in potential-aware abstraction. Both of these are conceptually clear improvements, and experiments showed in the large that the new algorithm outperforms the best prior abstraction algorithm with statistical significance.

A future direction would be to develop more accurate and/or faster heuristics for approximating EMD in our setting, or faster algorithms for computing exact EMD. One technique for achieving the latter would be to use a clustering algorithm that selects cluster centers that have lower dimensionality than the centers selected by k-means, since the best commercially-available algorithm for computing EMD is slow for points with high dimensionality. A promising approach is the k-medoids algorithm, which only uses data points as the cluster centers. However, k-medoids is significantly slower than k-means and it requires additional memory, so it is unclear whether it will be feasible to apply to poker or other large games. We would also like to extend our approach to the turn round as well, though this appears to be infeasible, even using our fast heuristic, due to the large number of turn hands that need to be clustered. One technique that may help is to sample a subset of the turn hands and perform clustering only over that subset.

We expect our algorithm to be applicable to games beyond no-limit Texas Hold'em (NLHE). We would expect it to lead to an even more significant performance improvement over the prior approach in certain other popular variants of poker, such as pot-limit Omaha Hold'em (aka PLO), since the strength of hands changes much more significantly between rounds than in Texas Hold'em, so taking the full trajectory of strength into account would be more important in that game. PLO also has a significantly larger state space than Texas Hold'em (in all rounds), and therefore we would expect abstraction to play a larger role than in NLHE. In particular, there are many more hands for the preflop and flop rounds in PLO than in NLHE, and we expect our approach to help the most when performing abstraction in the earlier rounds, since that is where there is the biggest difference between the distribution of hand strength assuming the final round is reached and the full trajectory of hand strength distributions in all future rounds.

We would also like to apply our algorithm to games outside of poker. It would be applicable to any large sequential game of imperfect information where information abstraction is necessary, and it would be especially useful for games where the strength of private information can change significantly between rounds, since the algorithm accounts for the full trajectory of strength over all future rounds.

7 Acknowledgments

This material is based on work supported by the National Science Foundation under grants IIS-1320620, CCF-1101668, and IIS-0964579, as well as XSEDE computing resources provided by the Pittsburgh Supercomputing Center. We would also like to thank Michael Johanson for providing us with the data used for Figures 1 and 2.

References

Arthur, D., and Vassilvitskii, S. 2007. k-means++: The advantages of careful seeding. In *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*.

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

Brown, N., and Sandholm, T. 2014. Regret transfer and parameter optimization. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*.

Elkan, C. 2012. Using the triangle inequality to accelerate k-means. In *Proceedings of the International Conference on Machine Learning (ICML)*.

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

Gilpin, A., and Sandholm, T. 2007a. Better automated abstraction techniques for imperfect information games, with application to Texas Hold'em poker. In *Proceedings of the International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*.

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

Gilpin, A., and Sandholm, T. 2008. Expectation-based versus potential-aware automated abstraction in imperfect information games: An experimental comparison using poker. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*. Short paper.

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

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 *Proceedings of the International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*.

Hawkin, J.; Holte, R.; and Szafron, D. 2011. Automated action abstraction of imperfect information extensive-form games. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*.

Hawkin, J.; Holte, R.; and Szafron, D. 2012. Using sliding windows to generate action abstractions in extensive-form games. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*.

Hoda, S.; Gilpin, A.; Peña, J.; and Sandholm, T. 2010. Smoothing techniques for computing Nash equilibria of sequential games. *Mathematics of Operations Research* 35(2). Conference version appeared in WINE-07.

Johanson, M.; Burch, N.; Valenzano, R.; and Bowling, M. 2013. Evaluating state-space abstractions in extensive-form games. In *Proceedings of the International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*.

Johanson, M. 2013. Measuring the size of large no-limit poker games. Technical report, University of Alberta.

Kroer, C., and Sandholm, T. 2014. Extensive-form game abstraction with bounds. In *Proceedings of the ACM Conference on Economics and Computation (EC)*.

Lanctot, M.; Waugh, K.; Zinkevich, M.; and Bowling, M. 2009. Monte Carlo sampling for regret minimization in extensive games. In *Proceedings of the Annual Conference on Neural Information Processing Systems (NIPS)*.

Lanctot, M.; Gibson, R.; Burch, N.; Zinkevich, M.; and Bowling, M. 2012. No-regret learning in extensive-form games with imperfect recall. In *Proceedings of the International Conference on Machine Learning (ICML)*.

Pele, O., and Werman, M. 2008. A linear time histogram metric for improved SIFT matching. In *Proceedings of the European Conference on Computer Vision*.

Pele, O., and Werman, M. 2009. Fast and robust earth mover's distances. In *Proceedings of the International Conference on Computer Vision*.

Sandholm, T., and Singh, S. 2012. Lossy stochastic game abstraction with bounds. In *Proceedings of the ACM Conference on Electronic Commerce (EC)*.

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*. London, UK: Springer-Verlag.

Waugh, K.; Schnizlein, D.; Bowling, M.; and Szafron, D. 2009a. Abstraction pathologies in extensive games. In *Proceedings of the International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*.

Waugh, K.; Zinkevich, M.; Johanson, M.; Kan, M.; Schnizlein, D.; and Bowling, M. 2009b. A practical use of imperfect recall. In *Proceedings of the Symposium on Abstraction, Reformulation and Approximation (SARA)*.

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