The Efficiency of the HyperPlay Technique Over Random Sampling

Michael Schofield and **Michael Thielscher**
School of Computer Science and Engineering
UNSW Australia
{mschofield, mit}@cse.unsw.edu.au

Abstract

We show that the HyperPlay technique, which maintains a bag of updatable models for sampling an imperfect-information game, is more efficient than taking random samples of play sequences. Also, we demonstrate that random sampling may become impossible under the practical constraints of a game. We show the HyperPlay sample can become biased and not uniformly distributed across an information set and present a remedy for this bias, showing the impact on game results for biased and unbiased samples. We extrapolate the use of the technique beyond General Game Playing and in particular for enhanced security games with in-game percepts to facilitate a flexible defense response.

Introduction

General Game Playing agents are expected to be able to play any game well with no prior knowledge or experience of the game. The game rules are declared in a Game Description Language (GDL) (Love et al. 2006), giving the roles in the game, the initial state of the game, the legal moves in any state of the game, the successor function for state transition, the terminal conditions and the reward structure for all roles.

In this field of research, games with imperfect information present a special challenge to the artificially intelligent agents. Such games provide limited information about the moves made by other roles leading to some uncertainty about the true state of the game. Therefore, the agent must reason across an information set in order to evaluate its move choices. This expands the search significantly over perfect-information games where all of the moves are known. In fact, an information set may be so large and the time constraints so tight that the agent may only be able to take a sample of an information set.

Perfect-information games are more popular among researchers as evidenced by the proliferation of GDL files (Tiltyard, Dresden) and competitions (Genesereth, Love, and Pell 2005) for perfect-information games. Imperfect-information research has been slow, with few games being converted to GDL-II and even fewer competitions.

The General Game Playing (GGP) community use the term "imperfect" to mean that some moves in the game may be hidden while the initial state of the game and the reward structure are known to all roles. The general AI community calls this "incomplete", and in the past we have done the same. However, there is enough awareness of this difference for us to use the GGP terminology.

The GDL-II specification (Thielscher 2010) augments the original GDL with the addition of percepts to give signals to agents, and a role for the random player[1]. If the percepts are limited then the agent may not be able to distinguish between several move histories in the game, giving rise to an information set (Schiffel and Thielscher 2014). Size and time constraints may force the agent to use all of its resources in an attempt to sample an information set in a very large search space,[2] resulting in little time for reasoning.

HyperPlay (Schofield, Cerexhe, and Thielscher 2012) overcomes this sampling difficulty by maintaining a continuously updatable bag of models of the game and uses this to provide a sample of an information set. As each move is made and percepts received, the models are updated by randomly adding to the history and so providing a new sample. When histories become invalid they are backtracked to a previously valid point and moved forward until a new sample is found.

Our motivation for this paper is proving the efficiency and effectiveness of the HyperPlay technique. Can it provide a sample of an information set more efficiently that a random sampling approach? Are there games where random sampling is impossible? Will the sample be uniformly distributed across an information set thereby providing an unbiased starting point for evaluation? If the sample is biased, can this be rectified?

In this paper we explore all of these questions using a basket of games that represent all aspect of "imperfect" information. We design experiments to expose the worst aspect of HyperPlay and look at the impact on competitive gameplay as well as remedies for any shortcomings. Additionally we supplement the games available in the GGP community by introducing GDL-II versions of two popular security games, the Transit game and the Border Protection game. Each game is reproduced in the GDL-II as part of the basket of games used to test out queries. Details of the conversion are presented later in the paper and a sample of the Transit game code is shown in Figure 1.

[1] The random role is "nature" in traditional game theory.

[2] In some game this is impossible within the constraints.

```
1  (role defender)
2  (role random)
3  (init (round 1))
4  (init (turn evader))
5  (init (location defender (cell 3 1)))
6
7  (<= terminal
8      (true (round 13)))
9  (<= (goal defender 100)
10     (true (evadercaught))
11     (true (returnedtobase)))
12
13 (<= (legal random (evaderat 1 ?y))
14     (file ?y)
15     (true (round 1)))
16 (<= (legal random (movesuccess ?p))
17     (movesuccess ?p)
18     (true (turn defender)))
19 (<= (legal defender (moveto ?x1 ?y1))
20     (true (location defender (cell ?x ?y)))
21     (adjacent ?x ?y ?x1 ?y1)
22     (true (turn defender)))
23 (<= (sees defender (evaderat ?x ?y))
24     (does random (evaderat ?x ?y))
25     (true (location defender (cell ?x1 ?y1)))
26     (adjacent ?x ?y ?x1 ?y1))
27 (<= (sees defender (movefail))
28     (does random (movesuccess 0)))
29
30 (<= (newlocation defender (cell ?x ?y))
31     (true (turn defender))
32     (not (does random (movesuccess 0)))
33     (does defender (moveto ?x ?y)))
34 (<= (next (turn defender))
35     (true (turn evader)))
36 (<= (next (location evader (cell ?x ?y)))
37     (true (turn evader))
38     (does random (evaderat ?x ?y)))
39 (<= (next (location defender (cell ?x ?y)))
40     (true (turn defender))
41     (newlocation defender (cell ?x ?y)))
42 (<= (next (location defender (cell ?x ?y)))
43     (not (true (turn defender)))
44     (true (location defender (cell ?x ?y))))
```

Figure 1: A sample of the GDL-II description of the Transit security game highlighting the key aspects of the game.

Background

In this section, we present the Game Description Language for imperfect-information games, showcasing the security games. We highlight differences in playing perfect-information and imperfect-information games, and look at information set sampling using HyperPlay and random approaches.

GDL-II

The General Game Playing agent requires a language to describe an arbitrary game with a set of rules. The Game Description Language (GDL) declares these rules with a formal syntax. The GDL was formalized by Love et al. (2006). Later Thielscher (2010) extended the language for imperfect-information games incorporating syntactic restrictions that ensure that every valid game description has a unique interpretation as a state transition system.

The enhanced language (GDL-II) has these additional properties:

- `(role random)` to act for nature; and
- `(sees ?role ?percept)` to provide signals.

In Figure 1 we see a partial GDL-II description of the Transit security game (Yin et al. 2012) adapted for GGP. Lines 1-2 show the roles[3], lines 3-5 show the initial state of the game, lines 7-8 show the terminal condition, lines 9-11 show one of the goal rules, lines 13-22 showcase the legal moves, lines 23-28 identify percepts, and lines 30-44 characterize the successor function.

Security games

Security games have been a topic of research recently with some success in real world applications, for example, public transport (Yin et al. 2012). Until recently these games become more accessible due to improvements in solving massive imperfect-information games (Bowling et al. 2015). It is this massiveness and the behavioral strategy that maps every action in every state to a probability that prevents on-line play. We take an adaptation of the Transit game and Border Protection (Bošanskỳ et al. 2015) and write an equivalent game in the GDL-II.

Transit Game sees an evader travel from left to right in the grid in Figure 2 while the defender start and finished at node S. This replicates the representation of Lisỳ, Davis, and Bowling (2016) which was solved using counterfactual regret minimization. We add the enhancement that the defender can get signals from adjacent nodes about the presence of the evader. This signal overlay allows for in-game response adjustments in two respects. Firstly there is the obvious "he's over there" which collapses an information set, and the more subtle "can't sees any evaders" which invalidates some models of the game causing re-sampling. It is the second aspect of signaling that is of interest in this paper.

Figure 1 shows a sample of the encoding of this game in the GDL. The random role makes moves for the evader, plus decides if the defender's moves are successful. The auxilliary relation `(newlocation defender (cell x y))` calculated in line 30-33 will fail if random chooses `(movesuccess 0)`, mimicking the behaviour in the original game.

Border Protection has an evader attempting to escape detection en-route with the protector covering the arrival points and receiving random signals from locations where no evader was detected. Again these negative signals invalidate some models of the game causing a re-sampling and a response adjustment. This version of the game wraps around so as to model the possibilities in global transportation.

[3]The role of the evader is played by random.

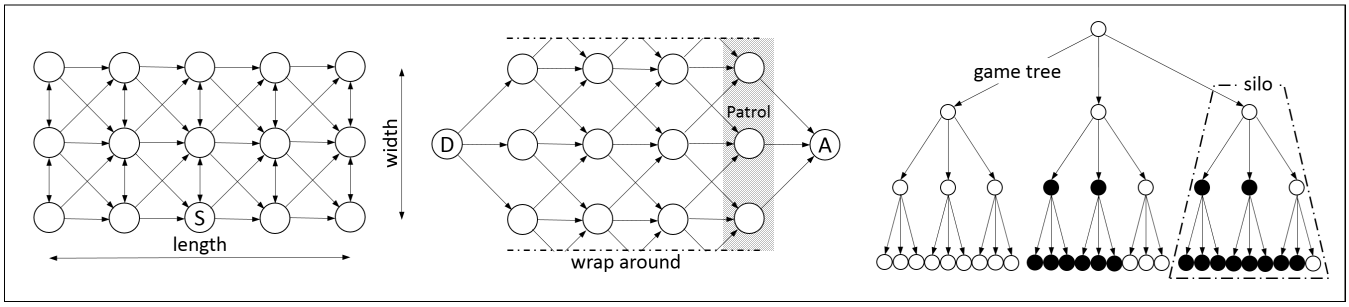

Figure 2: Left: Transit security game, Center: Border Protection security game, Right: Example of silos in the game tree.

Playing Imperfect-Information Games

Every GDL-II game description induces a game tree (Schiffel and Thielscher 2014) whose vertices map to states in the game and whose edges map to joint move vectors. Each vertex can be described by a set of fluents[4] as well as an ordered list of messages in the form of a history. Fluents can be identical in different states[5], but each vertex on the game tree has only one path (history) from the root (initial state). Therefore we work with histories not states to make evaluations.

An agent may have imperfect-information histories, not be able to distinguish one history from another and giving rise to an information set being defined as "the set of indistinguishable histories". Some histories in an information set are more likely than others and so we work with a weighted sample[6] based on the choices made along the path of the history. The agent grounds the unknown variables for a model of the game, and a sample of an information set.

Sampling an Information Set is used as the starting point for a forward search of the game tree. In order to generate the perfect-information histories required, each missing move must be substituted with one of the legal moves at the time, such that, all percepts arising from this substitution agree with the known percepts from the game (Schofield, Cerexhe, and Thielscher 2012)(Edelkamp, Federholzner, and Kissmann 2012). We use a deterministic approach starting from the initial state of the game working towards the current round. As the game progresses such valid substitutions become more difficult.

Move Selection maximizes the perceived payoff using a forward search of the game tree originating from each legal move choice. There are many techniques: UCT (Schäfer, Buro, and Hartmann 2008), MiniMax (Clune 2007), to name a few. In contrast to these, ISMCTS (Cowling, Powley, and Whitehouse 2012) overlays the individual searches into a single tree. However, like the others, ISMCTS[7] still guesses the hidden information. It is guessing the hidden information that is the focus of this paper.

[4]Things that are `(true ?)` in the game state.

[5]Imagine two board games with identical configurations, but different move histories.

[6]This is sometimes called a weighted particle filter (Veres and Norton 2001).

[7]We discuss ISMCTS in more detail after the conclusion.

HyperPlay

This technique maintains a bag of models of the game and updates them from one round to the next (Schofield, Cerexhe, and Thielscher 2012). This takes advantage of research into set sampling (Richards and Amir 2012) and particle system techniques (Silver and Veness 2010) where the sample of an information set is maintained from one round to the next. We show that the incremental update takes far less time and is less likely to fail than taking a new random sample for each round. If a percept is received that invalidates a model then the technique backtracks and explores the local subtree until a new model is found.

When a model is updated by substituting a legal move for the missing information a sample is created. In doing so, the sample is elevated to fact and the resulting evaluation suffers from the Strategy Fusion Error (Frank and Basin 1998). That is, the agent will not place any value on information gathering moves as it believes it already knows everything.

The Strategy Fusion Error has been overcome with HyperPlay-II (Schofield and Thielscher 2015), a nested player that employs an Imperfect-Information Simulation to evaluate the current set of legal moves. In turn, this technique suffers from poor scalability as it is a nested player and requires resources in the order of $O(n^2)$ (Schofield and Thielscher 2016).

Theoretical Analysis

In this section we present the theoretical basis for the experimental measurement.

Random Sampling

We define the probability of taking a valid sample randomly as the product of the probabilities of randomly choosing a valid legal move in each round of the game.

$$P(Valid(h_n)) \leq \prod_{R=1}^{n} P(Valid(\vec{a}_R)) \quad (1)$$

$$P(Valid(\vec{a}_R)) = |\{Valid(\vec{a}_R)\}|/|\{\vec{a}_R\}| \quad (2)$$

$$Valid(\vec{a}_R) = [\rho(\delta(s_R, \vec{a}_R)) = \rho(G_{R+1})] \quad (3)$$

We use h for history, R for round, $\vec{a}$ for joint move vector, s for state, G for game, $\rho()$ for percepts arising from actions, and $\delta()$ as the successor function.

Equation 1 multiplies the probabilities of successive valid joint-move vectors across rounds R. We use an inequality as a previously valid choice may be invalid at a later round. Equation 2 gives the probability of a valid move vector, while equation 3 defines a valid move vector as one where the precepts in the move implementation agree with the game currently being played.

Biased Samples

The HyperPlay technique advances each model by randomly substituting a legal move for missing information. If the move is invalid then it searches the local sub tree for a valid combination. In extreme cases the subtree is expanded beyond the local region until a new model is found. This gives rise to a shortcoming of this technique. That is, the game tree can be divided into a small number of subtrees based on the first legal move substitution. We call these subtrees "silos", as shown in Figure 2. Initially there will be an equal number of models in each silo. As the game progresses one silo may have only one viable history resulting in an over sampling as all of the models converge. We can compare sample histories to identify biased samples.

Uniformly Distributed Samples

We use a weighted particle filter, and so, some samples are more likely than others. However, *a priori* we must assume a uniform distribution across an information set.

When the sample size is smaller than the size of an information set it is difficult to measure the uniformity of the distribution, but when the sample size is much larger than the size of an information set then we can count the number of times each element of an information set is sampled and use Pearson's χ^2 measure for a uniform distribution as a measure of the probability that the observed distribution matches the expected distribution.

$$\chi^2 = \sum_{s \in H_R} (O_s - E_s)^2 / E_s \tag{4}$$

$$E_s = |M| / |H_R| \tag{5}$$

We use H for a set of histories, s for state, R for the round in the game and M for a bag of models.

Equation 4 sums the square of the difference between observed O_s and expected E_s sampling frequencies. The resulting statistic is converted to a probability via precalculated tables. In this case the expected value E_s is the same for each element in an information set, being the models per element.

Counting States Visited

It is common to measure a search of the game tree by counting the states visited or nodes touched. This is because the primary cost in traversing the game tree is the calculation of the successor function. By comparison storing and retrieving previously visited states from memory is low cost. Each state is counted as its node is touched in the game tree.

Design of Experiments

We outline the design of experiments to expose the shortcomings of HyperPlay and identify the impact on the agent.

Sampling Efficiency

The efficiency of the sampling process is tested by playing a batch of games and recording the states visited in each round when updating each of the models. Every round this number is written to a log file and the files collated. The statistic is examined and used to calculate the probability of successfully making a random selection as well as the cost of updating the model. The resources for each role are set so that it plays at well below the optimal level. This ensures good variety in the game-play and a broad base for the calculation of the statistic.

The HyperPlay process tests each move substitution in a random order, thus we can gain an accurate estimate of the probability in equation 2 by calculating the "first time" successes in that round.

Biased Samples

Bias is measured by playing a batch of games and logging the history footprint of each model for each round. The footprints are examined for repetition and a frequency chart is created. A Pearson's χ^2 test is then performed on the distribution and a probability value is calculated. The resources for each role are set so that it plays at well below the optimal level. This ensures good variety in the game-play and a broad base for the statistics.

Bias Remedies

These are examined by playing two batches of games, with and without remedies. The final scores for a designated player are averaged and reported along with a confidence interval. The resources for each player are set so that the player is competitive within a realistic time constraint based on the game complexity and the common competition times. In competition an agent would truncate its search to meet the time constraints, this would distort the statistic so we allowed the agent to complete the search for each round.

The first remedy was to inversely **weight** the results from each model based on its sample frequency. The mathematical adjustment would give each element of an information set an equal impact in the evaluation of outcomes. If an element was represented by 5 models then each model contributed only 20% of its outcomes to the evaluation process.

The second remedy was to re-**balance** the sample by replacing a more frequently sampled history with a less frequently sampled history so that each element was sampled the same number of times. Although this may seem to be the same as the first remedy it makes better use of the player's resources.

Roles

The roles were chosen to give meaningful results. In two-player turn-taking games we use the second player for the statistics as they receive imperfect-information first. In two-player simultaneous play we choose arbitrarily.

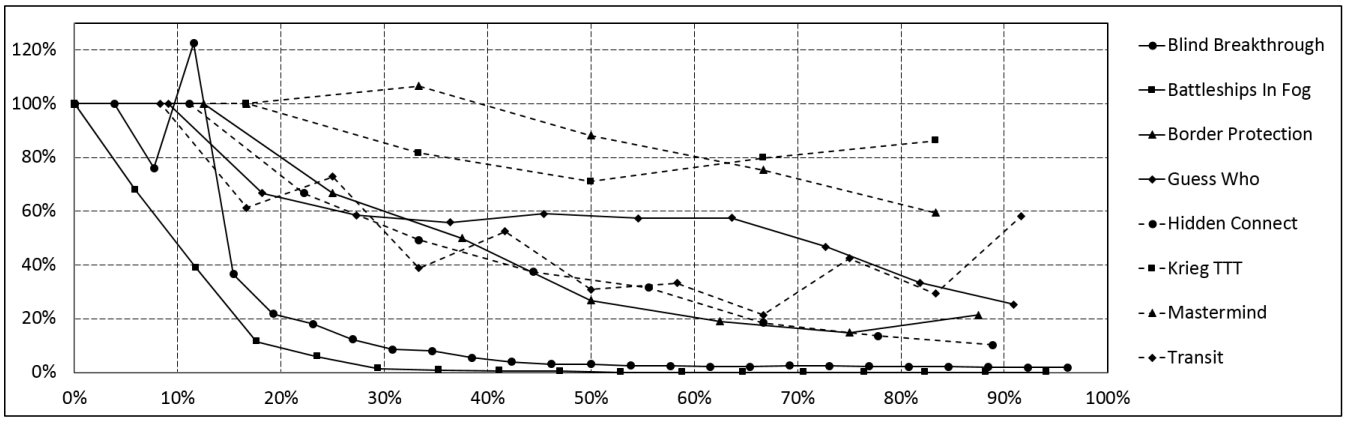

Figure 3: Cost of sampling using HyperPlay compared to performing a random sample, as measured by states visited.

Experimental Procedure

Because of variable run times on different hardware we report the number of states visited. When indicative times are given they are for computation performed by a single agent on an Intel Core i7-2600 @ 3.4GHz in a single thread.

Where appropriate a confidence level of 95% is used in a two-tailed calculation using the standard deviation for a binomial distribution. Where an average is statistically meaningless, a median and upper & lower quartiles are reported.

Batch sizes were calculated to give statistically meaningful results. Generally each experiment had a batch size of 1000 games, or 10,000 observations.

Games

The basket of games chosen for experiments was drawn from the games available within the GGP community, and from the newly converted security games. A variety of information imperfections are represented in the games. Cut down versions of the game are used, when possible, without loss of generality.

Battleships in Fog is a two-player, turn-taking, random start game with information gathering moves. It requires a HP-II (Schofield and Thielscher 2015) player to be played effectively.

Blind Breakthrough is a two-player, turn-taking blind variant of the Breakthrough game. There are no information gathering moves and it can be effectively played with a HyperPlayer using a tree search for move evaluation.

Border Protection is a two-player, turn-taking security game where one role is the random player. Percepts are provided to the defender to facilitate in-game response changes. There are no information gathering moves and can be effectively played with a HyperPlayer using a tree search for move evaluation.

Guess Who is a single-player mystery solving game where all moves are information gathering moves. It requires a HP-II player to be played effectively.

Hidden Connect is a two-player, turn-taking game that is a blind version of Connect4. There are no information gathering moves and it can be effectively played with a HyperPlayer using a tree search for move evaluation.

KriegTTT is a two-player, simultaneous move game that is a blind version of TicTacToe. There are no information gathering moves and it can be effectively played with a HyperPlayer using a tree search for move evaluation.

Mastermind is a single-player mystery solving game where all moves are information gathering moves. It is solved in the backtracking of invalid models by either HP or HP-II player. A well resourced player can achieve a binary search.

Transit is a two-player, turn-taking security game where one role is the random player. Percepts are provided to the defender to facilitate in-game response changes. There are no information gathering moves and it can be effectively played with a HyperPlayer using a tree search for move evaluation.

Case Study

Game variants and sizes have been chosen to prove (or disprove) the experimental objectives with a minimum of computational resources. However, there is value in extending the experimentation using one of the full scale versions of a common game in the form of a case study. We use Blind Breakthrough in the full 8x8 format to show how impractical random sampling can be. We use a sub-optimal HyperPlayer to ensure good variety in the game-play and a broad base for the calculation of the probability in equation 2.

The intention is to show that random sampling would become impossible within the normal time constraints, yet the HyperPlayer could successfully maintain a bag of models throughout the entire game. It should be noted that the HyperPlayer is capable of taking individual models off line if the backtracking process consumes too many resources. This is a design features for managing large search spaces.

Results

We present the experimental results along with comments that explain and highlight without drawing any conclusions.

Sampling Efficiency

In Figure 3 we show the HyperPlay cost of sampling compared to taking a random sample. The lines on the chart do

Game	Round	Q1	Median	Q3
Battleships In Fog	6	0.030	0.158	0.454
Blind Breakthrough	6	0.006	0.156	0.663
Border Protection	4	0.005	0.040	0.262
Guess Who	5	0.001	0.001	0.094
Hidden Connect	4	0.001	0.001	0.001
Krieg TTT	3	0.001	0.001	0.001
Mastermind	3	0.027	0.211	0.520
Transit	5	0.001	0.001	0.116

Figure 4: Probability of a uniformly distributed sample of an information set created by HyperPlay in mid game.

not represent any continuous function, they just connect results from the same game. The horizontal axis is a measure of completion of the game. When a game is 100% complete then it is terminal and no sampling is required.

The game-play is different for each game with some being turn-taking and others not, some games have watershed rounds where percepts collapse an information set. Thus, there is no rhyme or reason for the shape of the curves.

The experimental accuracy is not depicted on the chart. Each game was configured to provide approximately 10,000 observations for each data point on the chart.

The lone data point well above the 100% mark in Blind Breakthrough is when the black player initially encounters enemy pawns. The resulting re-sample is less efficient than taking new random samples.

Uniform Distribution

Figure 4 shows the probability that a sample performed by HyperPlay is uniformly distributed across an information set in the middle of a game. From Figure 3 we see that game play is not uniform, so the worst result (most biased) is shown from the mid-game rounds.

As one playout of a game is not the same as another it is not possible to average the results so we show a median and upper and lower quartile readings of the probability value from a Pearson's χ^2 test. The median value for Battleships in Fog of 0.158 infers that there is a 15.8% probability the sample is uniformly distributed.

Game	Round	Q1	Median	Q3
Battleships In Fog	12	0.004	0.058	0.150
Blind Breakthrough	11	0.246	0.998	1.000
Border Protection	7	0.001	0.001	0.004
Guess Who	10	0.121	0.399	0.792
Hidden Connect	8	0.001	0.001	0.001
Krieg TTT	5	0.001	0.167	0.992
Mastermind	5	0.413	0.899	1.000
Transit	11	0.996	0.999	1.000

Figure 5: Probability of a uniformly distributed sample of an information set created by HyperPlay in the end game.

Figure 5 shows a similar statistic for the end-game. Again the worst result is shown for the final rounds of the game.

It is worth noting that some samples are more uniform at the end of the game as the information set shrinks under certainty. Blind Breakthrough becomes a pawn swapping exercise towards the end game and nears certainty. Mastermind becomes certain as the binary search nears completion and the Transit game almost always becomes certain at the end when the evader is caught.

Game	Base	Weight	Balance
Battleships In Fog	82.0±2.4	85.3±2.2	82.8±2.3
Blind Breakthrough	52.5±3.1	51.4±3.1	53.2±3.1
Border Protection	51.0±3.1	49.2±3.1	51.0±3.1
Guess Who	67.8±2.9	68.6±2.9	69.4±2.9
Hidden Connect	37.2±3.0	36.6±3.0	36.0±3.0
Krieg TTT	51.6±3.1	50.9±3.1	51.7±3.1
Mastermind	92.8±1.6	93.3±1.6	93.8±1.5
Transit	72.2±2.8	74.6±2.7	73.0±2.8

Figure 6: Average performance of player using different remedies to unbalanced samples of an information set.

Remedy for Biased Samples

In Figure 6 we show the results of a batch of games played with different player configurations. The base case is two evenly matched players with no attempt to correct biased samples. The **weight** remedy reduce the weighting of a sample proportional to its repetition, and the **balance** remedy rebalances the sample every round. The mean values show a 95% confidence interval.

Round	$P(Valid(h_R))$	$sup\|\{h_R\}\|$	Active Models
1	100%	22	100%
2	100%	22	100%
-	-	-	-
16	1.19%	4.2 E+11	71.9%
17	0.73%	1.5 E+13	70.3%
-	-	-	-
32	$< 0.01\%$	6.0 E+23	50.0%
33	$< 0.01\%$	1.7 E+25	48.7%

Figure 7: Full sized Blind Breakthrough with the probability of randomly choosing a valid play history, an upper bound on the set size and the models still active.

Case Study

In Figure 7 we show the results from the full sized version of Blind Breakthrough. Both players were resourced just enough so as to exhibit a variety of game plays without making "stupid" moves. The results show an estimated upper bound on the set of play histories and the probability that a random history is valid in the game being played.

Also note that some models are taken off line by the HyperPlayer if they exceed 100,000 states visited in the backtracking stage. This is a design feature to prevent paralysis of the player. Such models can always be brought back on line if time permits.

Interpretation of Results

We present an interpretation of the results along with the main insights gleaned from the conduct of the experiments.

Sampling efficiency results show a significant reduction of the cost of sampling by using HyperPlay. The intuition here is that the cost of backtracking the local subtree will always be cheaper than starting each new sample attempt from the root node. While there will always be exceptions[8] the general rule is the longer and larger the game, the more efficient HyperPlay becomes.

Biased samples results show that in every game tested the sample became biased, with least mid-game bias in Mastermind with a 21% probability of an unbiased sample. However, by the end-game three of the games had given the agent enough percepts to allow it to re-sample in an unbiased way.

The biased sample is a genuine concern as many of the search techniques are mathematically predicated on a uniform random sample of an information set.

Remedies for biased samples was the hardest aspect of this research. That is, to find a repeatable, reproducible, realistic situation where the bias needs to be corrected in order to improve the agents performance. While it was possible to manipulate the game-play to create scenarios where choices were compromised by biased samples, these scenarios were so improbable as to have little impact on the average game.

By and large, the remedies for biased samples do not improve the agent's performance. However, the cost of both remedies is so small that it is prudent and mathematically reassuring to implement them.

The case study game is popular in GGP competitions, and so the results are very relevant to this work. In the 32nd round of a game the HyperPlayer could expect 1.3 models of a bag of 100 models to become inactive after each backtracking 100,000 states. The successful models took an average of 745 states to update, giving a total cost of 166,000 states to take the sample. Each valid random sample would cost more than $32/2/0.01\% > 160,000$ states. In this context, the random sampler would be completely ineffective taking only one sample for every 48 samples taken by the HyperPlayer.

This result is totally consistent with the cut down version of the game repoted in Figure 3 which shows a long term cost of 2.2% for HyperPlay over random.

[8]The first encounter of enemy pawns in Blind Breakthrough causes significant backtracking beyond the local subtree.

Conclusion

We conclude that HyperPlay is generally more efficient than a random search, and in some cases an order of magnitude more efficient. Clearly HyperPlay samples become biased, but with easy remedy. In some cases random sampling is impossible within the constraints of the game. In short, the technique is efficacious in games with imperfect information.

We expect this technique to be applicable in Artificial General Intelligence applications wherever an information set of indistinguishable action histories exists. Although the technique was developed within the context of General Game Playing it is not bound to that domain, or restricted by the Game Description Language. Any search that can be described using a connected, directed graph with a single root node that is acyclic in its undirected form will benefit from this technique.

Future Work

We previously mentioned Information Set MCTS (Cowling, Powley, and Whitehouse 2012). This represents an important advancement in imperfect-information game play with the possibility of significantly improving the move selection process for online players. Its primary focus is the forward search of the game tree and the evaluation of the move options. In this paper we focused on deterministic sampling of play histories (a backward looking search) and so we only used rudimentary random playouts for move evaluation.

Future work in this area would be our next priority and is likely to include:

- Can HyperPlay be coupled with ISMCTS to produce a superior player,
- Can ISMCTS overcome the Strategy Fusion Error and replace HyperPlay-II which is a "resource pig", and
- Can a player based on ISMCTS successfully play games like The Monty Hall Game which requires a weighted particle filter, and if not, can the HyperPlay ChoiceFactor be incorporated into such a player?

Acknowledgments

This research was supported by the Australian Research Council under grant no. DP150103034. The second author is also affiliated with the University of Western Sydney.

References

Bošanskỳ, B.; Jiang, A. X.; Tambe, M.; and Kiekintveld, C. 2015. Combining compact representation and incremental generation in large games with sequential strategies. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, 812–818. AAAI Press.

Bowling, M.; Burch, N.; Johanson, M.; and Tammelin, O. 2015. Heads-up limit holdem poker is solved. *Science* 347(6218):145–149.

Clune, J. 2007. Heuristic evaluation functions for general game playing. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 1134–1139.

Cowling, P. I.; Powley, E. J.; and Whitehouse, D. 2012. Information set monte carlo tree search. *IEEE Transactions on Computational Intelligence and AI in Games* 4(2):120–143.

Dresden game server. http://ggpserver.general-game-playing.de/ggpserver/. Accessed: 2016-08-30.

Edelkamp, S.; Federholzner, T.; and Kissmann, P. 2012. Searching with partial belief states in general games with incomplete information. In *Proceedings of the German Annual Conference on Artificial Intelligence (KI)*, 25–36.

Frank, I., and Basin, D. 1998. Search in games with incomplete information: A case study in using Bridge card play. *Artificial Intelligence* 100(1–2):87–123.

Genesereth, M. R.; Love, N.; and Pell, B. 2005. General game playing: Overview of the AAAI competition. *AI Magazine* 26(2):62–72.

Lisỳ, V.; Davis, T.; and Bowling, M. 2016. Counterfactual regret minimization in sequential security games. In *Proceedings of the AAAI Conference on Artificial Intelligence - Workshop on Computer Poker and Imperfect Information Games*.

Love, N.; Hinrichs, T.; Schkufza, D. H. E.; and Genesereth, M. 2006. General game playing: Game description language specification. Technical Report LG–2006–01, Stanford Logic Group.

Richards, M., and Amir, E. 2012. Information set generation in partially observable games. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Schäfer, J.; Buro, M.; and Hartmann, K. 2008. *The UCT algorithm applied to games with imperfect information*. Ph.D. Dissertation.

Schiffel, S., and Thielscher, M. 2014. Representing and reasoning about the rules of general games with imperfect information. *Journal of Artificial Intelligence Research* 49:171–206.

Schofield, M. J., and Thielscher, M. 2015. Lifting model sampling for general game playing to incomplete-information models. In *AAAI*, 3585–3591.

Schofield, M., and Thielscher, M. 2016. The scalability of the hyperplay technique for imperfect-information games. In *Proceedings of the AAAI Conference on Artificial Intelligence - Workshop on Computer Poker and Imperfect Information Games*.

Schofield, M.; Cerexhe, T.; and Thielscher, M. 2012. HyperPlay: A solution to general game playing with imperfect information. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 1606–1612.

Silver, D., and Veness, J. 2010. Monte-Carlo planning in large POMDPs. In *Proceedings of the Annual Conference on Neural Information Processing Systems (NIPS)*, 2164–2172.

Thielscher, M. 2010. A general game description language for incomplete information games. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 994–999.

Tiltyard. http://tiltyard.ggp.org/. Accessed: 2016-08-30.

Veres, G., and Norton, J. 2001. Weighted particle filter for state estimation. In *Proceedings of the IASTED International Conference: Control and Applications, Banff, Canada*, 115–120.

Yin, Z.; Jiang, A. X.; Johnson, M. P.; Kiekintveld, C.; Leyton-Brown, K.; Sandholm, T.; Tambe, M.; and Sullivan, J. P. 2012. Trusts: Scheduling randomized patrols for fare inspection in transit systems. In *Proceedings of the Innovative Applications of Artificial Intelligence Conference*.