General Game Playing with Imperfect-Information Appendix

Michael Schofield mschofield@cse.unsw.edu.au
Michael Thielscher mit@cse.unsw.edu.au
School of Computer Science and Engineering,
University of New South Wales,
Sydney Australia

Introduction

This appendix supports the article General Game Playing With Imperfect Information with an expansion of the Game Description Language, supplementary nomenclature, explanations of the games featured in this research, and experimental results from the several stages of development.

In the main article we draw on the General Game Playing research for perfect-information games and extends it to imperfect-information games. We take the simple technique of using a sample of the player's information set[1] and confirm that only a weighted sample will provide the correct results. We then devise a formulation for calculating weightings for the particle filter from partial probabilities using Ockham's Razor.

The idea of sampling an information set is then extended to search spaces where such sampling would normally be intractable. We offer a technique (HyperPlay) that maintains a bag of samples from one information set to the next in a computationally efficient and effective manner. This allows us to generate samples of an information set that would otherwise be impossible within the time budget given for each move. The technique is proven to be sound and complete for all imperfect-information games in General Game Playing. A formalism for an implementation of the technique is presented and tested to reveal its strengths, weaknesses and its limitations. Experimental data is presented to support the conclusions.

1. Sometime this is referred to as a "particle filter".

Appendix A. Game Description Language

A.1 GDL-II Keyword Description

The General Game Player requires a formal language that allows an arbitrary game to be specified by a full set of rules. The Game Description Language (GDL) is used to specify a set of declarative statements (rules) with a programming-like syntax. Originally designed for games of perfect-information, the GDL was formalized by Genesereth, Love, and Pell (2005) and later enhanced to include games of imperfect-information (Thielscher, 2010), as characterized by the special keywords listed in Table 1.

`(role ?r)`	`?r` is a player	`(legal ?r ?m)`	`?r` can do `?m`
`random`	the random player	`(does ?r ?m)`	player `?r` does move `?m`
`(init ?f)`	`?f` holds in initial state	`(sees ?r ?p)`	`?r` perceives `?p`
`(true ?f)`	`?f` holds in current state	`(terminal)`	current state is terminal
`(next ?f)`	`?f` holds in next state	`(goal ?r ?v)`	`?r` gets payoff `?v`

Table 1: GDL-II keywords

GDL-II comes with some syntactic restrictions[2] that ensure that every valid game description has a unique interpretation as a state transition system, as follows.

1. The **players** in a game are determined by the derivable instances of `(role ?r)`.
2. The **initial state** is the set of derivable instances of `(init ?f)`.
3. For any state S, the **legal moves** of a player `?r` are determined by the instances of `(legal ?r ?m)` that follow from the game rules augmented by an encoding of the facts in S using the keyword `true`. Game play is synchronous[3][4] and states are updated by joint moves with one move by each player `?r`.
4. The **next** position after a joint move is taken in state S is determined by the instances of `(next ?f)` that follow from the game rules using the keywords `does` and `true`.
5. The **percepts** a player `?r` gets as a result of a joint move being taken in state S is determined by the derivable instances of `(sees ?r ?p)` using `does` and `true`.
6. The rules for `terminal` and `goal` determine whether a given state is **terminal** and what the players' **goal** values are in this case.

2. For details, we refer to the language specifications (Love, Hinrichs, Schkufza, & Genesereth, 2006; Thielscher, 2010).
3. Synchronous means that all players move simultaneously. Turn-taking games are modelled by allowing players only one legal move without effect (such as `noop`) if it is not their turn.
4. "NoOp" is short for No Operation.

Appendix B. Definitions

Definition 1. Let $G = \langle S, s_0, R, A, \lambda, P, \rho, \upsilon, \delta \rangle$ be an imperfect-information game given by a valid GDL-II description, then:

1. S is a set of states, disjoint decision and terminal states $S = D \uplus T$;
2. $s_0 \in S$ is the initial state of the game;
3. R is a set of roles in the game;
4. A is the set of all moves (actions) in the game;
5. $\lambda : D \times R \to 2^A$ is a function giving a set of legal moves for $r \in R$ in $d \in D$;
6. P is a set of all percepts in the game;
7. $\rho : D \times A^{|R|} \times R \to P$ is a function giving the percept (with multiple percepts conjoined and *null* as the empty percept) for role $r \in R$ resulting from enacting a joint move in a decision state $d \in D$;
8. $\upsilon : T \times R \to \mathbb{R}$ is the payoff function on termination; and
9. $\delta : D \times A^{|R|} \to S$ is the successor function. □
   Supplementary nomenclature used hereafter are detailed in Appendix B.

We supplement Definition 1 with the following syntax and semantics. We use:

- $n \in \mathbb{N}$ is the number of joint moves enacted in a game that is in progress,
- $s_t \in S$ for the current state of a game in progress,
- subscript $r \in R$ to mean "belonging to" a role,
- $a_r \in \lambda(d, r) \subseteq A$ as a legal move for role $r \in R$ in decision state $d \in D$,
- $\lambda : D \to 2^{A^{|R|}}$ as an overload[5] of the function λ to return the set of legal move vectors $\vec{a}$ in $d \in D$, where $\vec{a} = \langle a_1 ... a_{|R|} \rangle \in \lambda(d)$ with one move for each role,
- $\langle \vec{a}_{-r}, a_r \rangle = \langle a_1 \dots a_r \dots a_{|R|} \rangle$ as a move vector containing a specific move a_r,
- $p_r = \rho(d, \vec{a}, r)$ as the percept received by role $r \in R$ after succession,
- $\rho : D \times A^{|R|} \to P^{|R|}$ as an overload of the function ρ to return a percept vector $\vec{p}$ in $d \in D$ enacting the move vector $\vec{a}$, where $\vec{p} = \langle p_1 ... p_{|R|} \rangle = \rho(d, \vec{a})$, and
- $s = \delta(d, \vec{a})$ as the progression of the game in $d \in D$ enacting move vector $\vec{a}$.

Definition 2. A game $G = \langle S, s_0, R, A, \lambda, P, \rho, \upsilon, \delta \rangle$ given by a valid GDL-II description induces a *game tree*, which is a connected, acyclic graph $\mathbb{G} = (\mathbb{V}, \mathbb{E})$ with a single root node for the initial game state and where the edges are determined by the joint legal moves in non-leaf nodes while leaves correspond to terminal game states. We will use the following definitions:

- *state* $: \mathbb{V} \to S$ is a function that maps from a node (vertex) $\mathrm{v} \in \mathbb{V}$ in the game tree to the corresponding game state $s \in S$; and
- *moves* $: \mathbb{E} \to A^{|R|}$ is a function that maps from an edge $\mathrm{e} \in \mathbb{E}$ in the game tree to the corresponding joint move $\vec{a}$;
- $\vec{\mathrm{e}}^n$ is a unique path of edges $\langle \mathrm{e}_1, \mathrm{e}_2, ... \mathrm{e}_n \rangle$ beginning at the single root node v_0, corresponding to the joint moves enacted in a game; and
- *node*$^i : \mathbb{E}^{\mathbb{N}} \times \mathbb{N} \to \mathbb{V}$ is a function that returns the i^{th} node along a path e^n. □

5. We overload functions as the need arises. We are careful to declare the overload at the time.

Definition 3. Let $G = \langle S, s_0, R, A, \lambda, P, \rho, \upsilon, \delta \rangle$ be an imperfect-information game given by a valid GDL-II description and $\mathbb{G} = (\mathbb{V}, \mathbb{E})$ be the induced game tree, and a play message be a move and percept tuple associated with an edge on the induced game tree and a history be a sequence of such edges. Then:

- M is the set of all play messages, $m = \langle a, p \rangle \in M$, for each role $m_r = \langle a_r, p_r \rangle$ where a_r is the move enacted in decision state d and $p_r = \rho(d, \vec{a}, r)$;
- $\mathcal{H}$ is the set of all histories in the game with $h \in \mathcal{H}$ being a history of $\vec{m}^k$; [6]
- $\xi : \mathcal{H} \times R \to \mathcal{H}$ gives the imperfect-information history as seen by role $r \in R$, and
- $I_{r,n} \subseteq \mathcal{H}$ is the general form for an information set for role $r \in R$ in round n. □
  Supplementary nomenclature used hereafter are detailed in Appendix B.

We supplement Definition 3 with the following syntax and semantics. We use:

- $\vec{m} = \langle m_1, ...m_{|R|} \rangle$ as a play message vector, one message for each role,
- $\vec{m_r} = \langle null, ...m_r, ..., null \rangle$ as an imperfect play message vector for role $r \in R$,
- $\vec{m_r}^n = \langle \vec{m}_{r1}, ...\vec{m}_{rn} \rangle$ as an imperfect history for role $r \in R$ in round n,
- $\xi : \mathbb{V} \times R \to \mathcal{H}$ as an overload of the function ξ that give an imperfect history for a node for role $r \in R$, with $\xi(\mathrm{v}_t, r)$ being an imperfect history of the game, and
- $node^i : \mathcal{H} \times \mathbb{N} \to \mathbb{V}$ is an overload of the function $node^i$ for histories.

Definition 4. Let $G = \langle S, s_0, R, A, \lambda, P, \rho, \upsilon, \delta \rangle$ be an imperfect-information game given by a valid GDL-II description and $\mathbb{G} = (\mathbb{V}, \mathbb{E})$ be the induced game tree and each role have a move selection policy expressed as a probability distribution across all moves in all decision states. We use the following definitions:

- Π is the set of all move selection policies $\pi \in \Pi$,
- $select : \Pi \times D \times R \to \phi(A)$ is a move selection function, as a probability distribution across all moves, and
- $play : S \times \Pi^{|R|} \to \phi(T)$ is the playout of a game from any state to termination according to the given move selection policies, expressed as a probability distribution across all terminal states. □
  Supplementary nomenclature used hereafter are detailed in Appendix B.

We supplement Definition 4 with the following syntax and semantics. We use:

- $\vec{\pi} = \langle \pi_1, .., \pi_{|R|} \rangle$ as a tuple of move selection policies, one for each role, and
- $\phi(A) = select(\pi_r, d, r)$ is the move selection probability for $r \in R$ in $d \in D$.

6. We depart from the convention of a history being only actions a^k or edges e^k as we need to deal with imperfect histories, where the percepts provide additional information that is not in the state but aids in partitioning nodes into information sets,

Appendix C. Games

Exploding Bomb

```
(succ 0 1)
(succ 1 2)
(succ 2 3)
(color red)
(color blue)

(role agent)
(role random)

(init (round 0))

(<=(legal random (arm ?c))
   (color ?c)
   (true (round 0)))
(<=(legal random noop)
   (not (true (round 0))))
(<=(legal agent noop)
   (true (round 0)))
(<=(legal agent ask)
   (true (round 1)))
(<=(legal agent wait)
   (true (round 1)))
(<=(legal agent (cut ?c))
   (color ?c)
   (true (round 2)))


(<=(sees agent ?c)
   (does agent ask)
   (true (armed ?c)))

(<= (explodes)
    (does random (cut ?c))
    (not (true (armed ?c))))
(<= (next (round ?n))
    (true (round ?m))
   (succ ?m ?n))
(<= (next (armed ?c))
    (does random (arm ?c)))
(<= (next (armed ?c))
    (true (armed ?c)))
(<= (next (score 90))
    (does explodes ask))
(<= (next (score 100))
    (does explodes wait))
(<= (next (score ?s))
    (not (explodes)))
(<= (next (score 0))
    (explodes))

(<= (terminal)
    (true (round 3)))
(<=(goal agent ?s)
   (true (score ?s)))
```

Figure 1: GDL-II description of the Exploding Bomb game.

Appendix D. HyperPlayer Experimental Results

We recapitulate the experimental results presented by Schofield, Cerexhe & Thierlscher in their original paper (Schofield et al., 2012).

D.1 Player Resources

There are two unconstrained variables in our move selection policy. One, the number of models (HyperGames) in our bag, and two, the number of playouts used to calculate the average utility. Initially we set these two variables equal in value and express this in the form "16x16", meaning 16 HyperGames in the bag and 16 playouts averaged to give move utility.

When we consider the quantity of resources, we use the term "fully resourced player" to mean that the addition of more resource would not improve performance within a 99% confidence interval.

D.2 Confidence Level

For each game variant shown in this section, an experiment was run one thousand times and the average results reported. For the two player games a result of zero indicated that the opponent scored the maximum payoff in every game, and a result of 100 indicated that the HyperPlayer scored the maximum payoff in every game.

In order to validate out technique, the data should show that the results tend towards a limiting value for fully resourced players. That is, some large number of models of the game can be substituted for perfect-information about the game.

A confidence level of 99% was used in a two-tailed calculation using the standard deviation for a binomial distribution.

D.3 Equipment

Experiments were designed for high throughput computing using the Condor computing array at the School of Computer Science and Engineering at the University of New South Wales. The experiments were broken up into small batch runs and scheduled across (approx) 200 dual core Intel PCs situated throughout the School. The HyperPlayer was given sufficient resources to compute each move in less than 15 seconds.

D.4 Standardized Opponent

In each of the two player games the HyperPlayer opposed a Cheat, a HyperPlayer with access to the true game, who maintains HyperGames that are the game (instead of models of the game). The Cheat was fully resourced so that it made the best move choices within the limitations of the move selection process.

D.5 Games

Monty Hall In addition to the rules already presented, we varied the number of initial doors between three, four, and five doors. The host always opens all but two doors.

Krieg-TicTacToe A variant of traditional TicTacToe where the opponent's pieces are hidden. Players move simultaneously and the winning length was fixed at four in a row. We varied the board size between 4x4, 5x5, and 6x6 squares. Players are given feedback via percepts as to whether their move, and their opponent's move, were successful.

Blind Breakthrough A variant of chess where each player starts with two rows of pawns on their side of the board. Players take turns, trying to 'break through' their opponent's ranks to reach the other side first. Opponent's pieces are hidden, as with Krieg-TicTacToe. The board size was varied between 5x5, 6x6, and 7x7 squares and players were given equal opportunities to go first.

D.6 Results

D.6.1 Particle Filter Weighting

We can validate the weightings applied to the HyperGames by playing the Monty Hall game. If our formulation is correct then the maximum long term score should be equal to $100 \cdot (1 - 1/No\_Of\_Doors)$, otherwise the maximum long term score should be 50. Clearly, the former is demonstrated in Figure 2.

D.6.2 Monty Hall

As expected, the adequately resourced HyperPlayer was able to achieve an average payoff appropriate for the number of doors in the game.

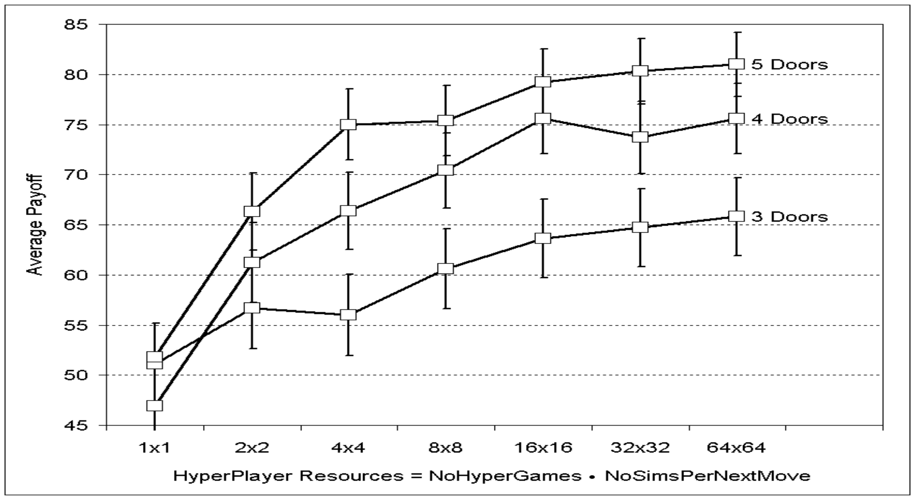

Figure 2: Monty Hall results, validating the weighting used in the particle filter.

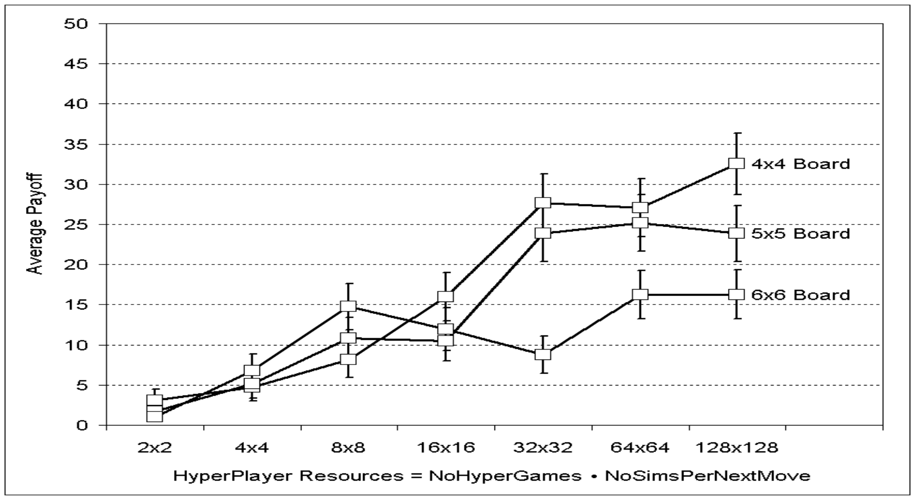

Figure 3: Krieg-TicTacToe results

D.6.3 Krieg-TicTacToe

These experiments showed steady improvement in performance as HyperPlayer resources increased. The limiting values appear to be well short of the 50% mark, especially on the larger board. On inspection of the play messages, it could be seen that the reduced number of percepts relative to the game duration gave the HyperPlayer very little information to assist in narrowing its search. In fact, the Cheat often won the game before the HyperPlayer could establish an accurate model of the board.

D.6.4 Blind Breakthrough

The results of the Blind Breakthrough experiments show clear improvement in performance as resources increase and approach a limiting value. As the number of HyperGames increases and the number of simulations per move increases the HyperPlayer is able to match the Cheat's performance with neither player having the advantage. In the 5x5 results, the percepts were sufficient for the HyperPlayer to maintain models that were very close to the true game. This allowed the HyperPlayer to perform as if it had perfect-information.

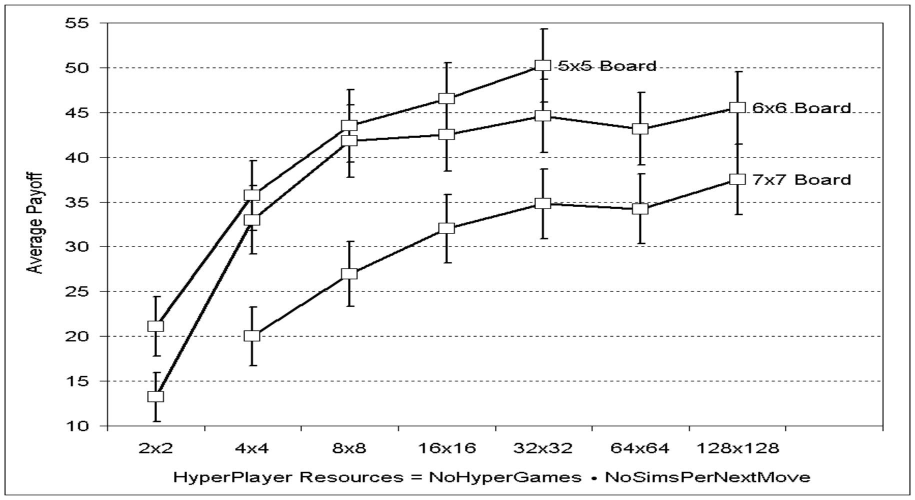

Figure 4: Blind Breakthrough results

Appendix E. Efficiency of HyperPlayer Results

We recapitulate the experimental results presented by Schofield and Thielscher in their original paper (Schofield & Thielscher, 2017).

E.1 Design of Experiments

A batch of games is played while 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 to ensures good variety in the game-play and a broad base for the calculation of the statistic.

E.2 Game Play

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. For example, the Blind Breakthrough would normally be played on a 8x8 board, but a 5x5 version is used to examine sampling efficiency.

E.3 Roles

The roles were chosen to give meaningful results. In two player turn-taking games the second player is used for the statistic as they receive imperfect-information first.

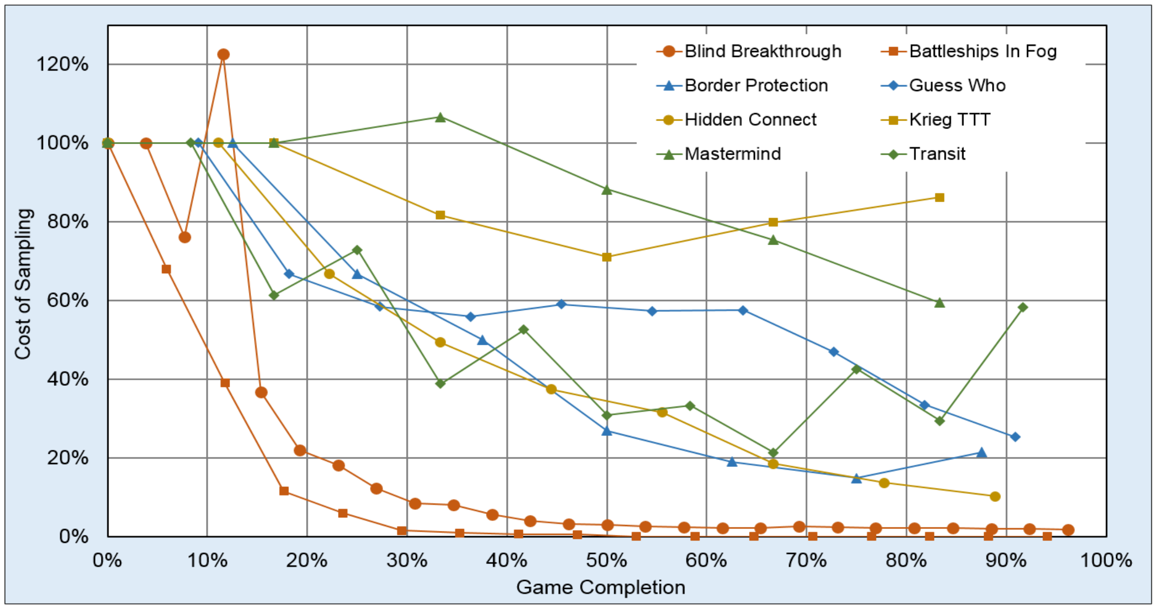

Figure 5: Cost of sampling using HyperPlay compared to performing a random sample, as measured by states visited.

E.4 Measuring Performance

The HyperPlay process tests each move substitution in a random order, thus it is possible to gain an accurate estimate of the probability in equation 16 by calculating the "first time" successes in that round.

The game-play is different for each game with some being turn-taking others not, some games have watershed rounds where percepts collapse an information set. Thus, there is no rhyme or reason 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 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 the general rule is the longer and larger the game, the more efficient HyperPlay becomes. 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.

E.5 Probability of Uniform Sample

Table 2 shows the probability that a sample performed by HyperPlay is uniformly distributed across an information set in the middle of a game. From Figure 5 it can be seen that game play is not uniform, so the worst result (most biased) is shown from the mid-game rounds.

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

Table 2: Probability of a uniformly distributed sample of an information set created by HyperPlay in mid game.

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

Table 3: Probability of a uniformly distributed sample of an information set created by HyperPlay in the end game.

As one playout of a game is not the same as another it is not possible to average the results so a median and upper and lower quartile readings of the probability value from a Pearson's χ^2 test are shown. The median value for Battleships in Fog of 0.158 infers that there is a 15.8% probability the sample is uniformly distributed. Some median values are extremely low at 0.001, or 0.1% probability of a uniform distribution.

Table 3 shows a similar statistic for the end-game. Again, the worst result is shown for the final rounds of the game.

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

Table 4: Average performance of player using different remedies to unbalanced samples of an information set.

E.6 Remedy for Biased Samples

Table 4 shows the results of a batch of games played with different player configurations. The base case is two evenly matched player with no attempt to correct biased samples. The **weight** remedy reduce the weighting of a sample proportional to its repetition so that each unique sample is given equal consideration before the application of the particle filter weighting, and the **balance** remedy re-balances the sample every round by replacing the oversampled play history with an under-sampled play history. The mean values show a 95% confidence interval.

Appendix F. HyperPlayer-II Experimental Results

We recapitulate the experimental results presented by Schofield, Cerexhe & Thierlscher in their original paper (Schofield & Thielscher, 2015). The "original technique" refers to a HyperPlay and the "new technique" refers to HyperPlay-II.

The experimental design, confidence level and equipment are all the same as Appendix D.

F.1 Player Resources

In each experiment, the player resources were varied to demonstrate the performance as a function of resources. Care was taken to ensure that each player had equal resources when making a move selection. This was achieved by setting each player's parameters such that they would visit a similar number of states each round. A player resource index of zero represents random decision-making and serves to provide a basis for improvement.

F.2 Games

Games played at the Australasian Joint Conference on Artificial Intelligence 2013 were used as inspiration for the experiments to validate the claim that the new technique correctly values moves that seek or protect information. The conference organizers specially designed games that would challenge the state of the art of GDL-II players to encourage research and development in this field.

Exploding Bomb This simple game commences with the *random* player choosing a red or blue wire to arm a bomb. Next, the agent may choose whether to ask which wire was used. Asking carries a cost of 10%. Finally, the agent must cut one wire to either disarm, or detonate, the bomb. We use this game as our running example.

Spy vs. Spy A simple variant of the Exploding Bomb game reverses the information flow. In this version the arming agent—who chooses which wire arms the bomb—also decides whether to tell the other player which wire to cut. Withholding this information carries a penalty of 20%. This tests the value a player places on giving away information.

Number Guessing The agent must guess a random number between 1 and 16. It can ask if the number is 'less than X', or can announce it is 'ready to guess', then guess the number. The score is discounted by time after the first 5 moves.

Banker and Thief This game tests a player's ability to keep secrets, ie. to value withholding information. There are two banks, a banker and a thief. The banker distributes ten $10 notes between the two banks. The Banker scores all the money left in his bank at the end of the game, except his bank has a faulty alarm system. The thief can steal all the money from the faulty bank, if they can identify it. The challenge for the banker is not to reveal the faulty bank by over-depositing.

Battleships In Fog This turn-taking, zero-sum game was designed to test a player's ability to gather information and to be aware of information collected by its opponent. Two battleships occupy separate grids. A player can fire a missile to any square on the opponent's grid, move to an adjacent square, or scan for their opponent. If they scan, they will get the exact location and their opponent will know that they have been scanned.

F.3 Results

F.3.1 Exploding Bomb

The original player never asks the question in this game since it thinks it already knows the answer (due to superficial agreement of its samples) and so can avoid the modest penalty. In contrast, the new player correctly identifies that asking the question gives the best expected outcome.

round	agent does	HyperPlay	HyperPlay-II
1	ask	45.04 ± 0.09	**90.00 ± 0.00**
1	wait	**49.98 ± 0.10**	49.91 ± 0.64
2	cut unarmed	**49.40 ± 1.19**	0.00 ± 0.00
2	cut armed	**50.60 ± 1.19**	**90.00 ± 0.00**

Table 5: Experimental score calculations during the Exploding Bomb decision-making process. The bold scores indicate the chosen actions.

F.3.2 Spy vs. Spy

Table 6 shows experimental results in the form of calculated expected outcomes. The original player always tells to avoid the penalty. The new player recognizes that hiding this information yields a better expected outcome.

arming agent does	HyperPlay	HyperPlay-II
arm blue and tell	**60.00 ± 0.15**	20.00 ± 0.00
arm red and tell	**60.04 ± 0.14**	20.00 ± 0.00
arm blue and hide	39.98 ± 0.16	**40.36 ± 1.22**
arm red and hide	39.99 ± 0.14	**39.45 ± 1.33**

Table 6: Expected score calculations for the arming agent in round one of the Spy vs. Spy decision-making process. The bold scores indicate the chosen actions.

F.3.3 Number Guessing

The original player always announces it is ‘ready to guess’, but then guesses randomly resulting in a 6.25% chance of guessing correctly. The new player only guesses the number when all playouts agree on the result. Binary search plays perfectly here, guessing after four questions. In Figure 6 the new player approaches this score.

F.3.4 Banker and Thief

Figure 7 shows that the original technique adopts a greedy policy and places \$100 in its bank, only to have it stolen. The new technique, adequately resourced, will deposit \$40 of the money in its bank, relying on a greedy thief to attempt to steal the \$60 in the other

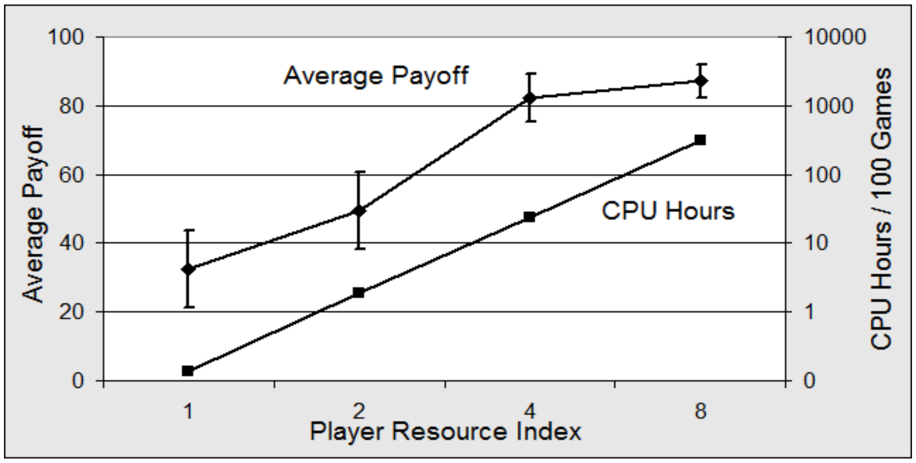

Figure 6: The NumberGuessing Results for HyperPlay-II.

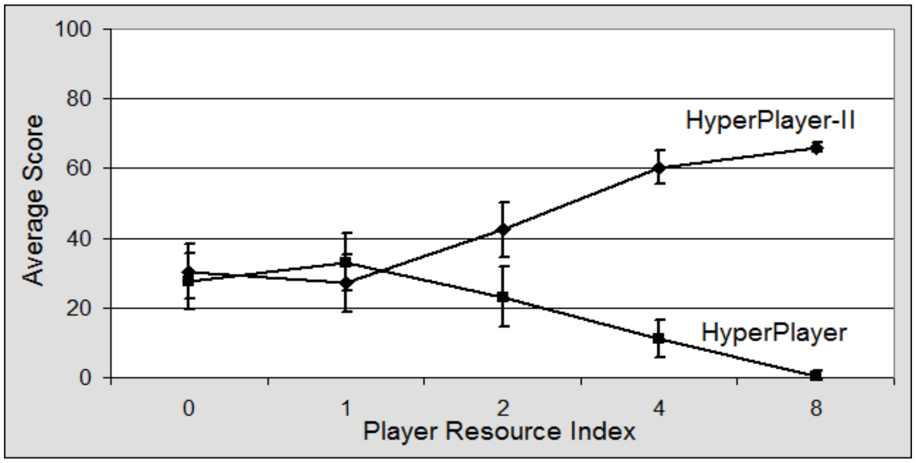

Figure 7: The Banker and Thief results.

bank. The new technique reaches optimal $avg(40 + 100)$ at resource index of eight as it correctly models both roles.

F.3.5 Battleships In Fog

The original player sees no value in scanning as all of the samples ‘know’ where the opponent is. It does not value moving after being scanned as it thinks its opponent always knows where it is. Its only strategy is to randomly fire missiles giving it a 6.25% chance of a hit on a 4x4 board. The new player will scan for the opponent and fire a missile.

A resource index of four is sufficient for the new player to dominate the old in this turn-taking game: HyperPlay has a 9.4% chance of winning with a random shot (12.5% if it goes first, half that if it plays second).

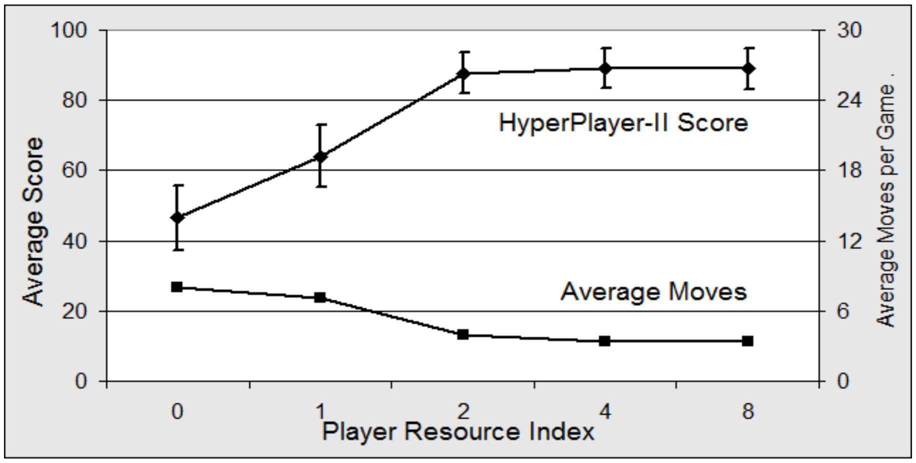

Figure 8: The Battleships In Fog results.

Appendix G. Imperfect-Information Game Topology

G.1 Imperfect-Information Games

In the General Game Playing domain the rules of the game, and hence, the reward structure is fully known to each player. What is not known are the moves made by other players. A player only receives percepts from the game controller according to the rules of the game. Therefore, we look at the variations that can occur in the topology of a game.

G.1.1 Imperfect Move Information

This is perhaps the simplest type of game, where a player must compete with another player whose moves are hidden, only receiving some clue about the game play from time to time. Whilst this is true for almost all imperfect-information games there are some games where the moves are the only thing that is hidden. Many games fit this category including board games that have been adapted for imperfect-information. We use Hidden Connect to represent this type of game.

G.1.2 Imperfect Initial Information

This is also one of the simpler type of game, where a player must search out the solution to a challenge that starts with some missing information. Often the random player (nature) makes some hidden move to begin the challenge. Many games fit this category. We use the game Mastermind to represent this type of game.

G.1.3 Information Purchasing

This type of game test the player's ability to value information gathering moves. Generally, the player can choose between asking a question about the state of the game and attempting to meet the criteria for success. Information gathering moves incur a cost through a reduction in the final score. There are fewer of these games played in General Game Playing and in the community. We use the Number Guessing game to represent this type of game.

G.1.4 Hidden Objectives

This type of game test the player's ability to identify their opponent's success criteria, or to hide their own success criteria. There are very few of these games played in General Game Playing and in the community. We use Banker and Thief from the GGP competition at the Australasian Joint Conference on Artificial Intelligence to represent this type of game.

G.1.5 Tactical Information Valuing

This type of game test the players ability to correctly value information in terms of a tactical advantage in the game play, both the cost of collecting information and the cost of keeping secrets. There is no direct cost for information gathering moves, but there is a tactical cost/benefit in terms of the expected outcome of the game. There are very few of these games played in General Game Playing and in the community. We use Battleships in the Fog as played in the GGP competition at the Australasian Joint Conference on Artificial Intelligence to represent this type of game.

Appendix H. HyperPlayer Scalability Experimental Results

We recapitulate the experimental results presented by Schofield & Thielscher in their workshop paper (Schofield & Thielscher, 2016).

The experimental design, confidence level and equipment are all the same as Appendix D.

H.1 Game Play

For single player games, there is no issue with game play, but with two player games we need a consistent opponent to make some useful measurements. To this end we instantiate an opponent who uses the HP technique and is adequately resourced to be competitive. Since the measurements are comparative, the experiments will not be overly sensitive to the performance of the opponent.

H.2 Player Configuration

For the HP player we can alter two resource parameters: the size of the bag of models,[7] and the number of playouts per legal move. We say that HP(4, 2) maintains 4 models of the game and uses two playouts for each legal move to make a move choice. That would mean in a game with a branching factor of 10 and a playout depth of 10 there would be 800 states visited in make a move choice, and 4,400 states visited in playing a game.

The HP-II player is twice as complex as it is a level 2 nested player, so we can alter four resource parameters. For example, we would say that HP-II(4, 2, 4, 2) was equivalent to HP(4, 2, HP(4 ,2)) which significantly increased the number of states visited from 800 to 352000[8] for a move choice.

Preliminary experiments were conducted to find the best configuration of both players for each game. The intent was to show the best performance for each player in terms of maximizing the score and minimizing the number of states visited. Once that configuration was found we then used multiples of 2 to produce characteristic curves presented in the results.

H.3 Measuring Performance

It is common in GGP to simulate the game play of a competition by giving each player a time budget for each move. However, there have been very few GGP-II competitions and the idea of a time budget has less meaning. Also, a time budget is very dependent on the hardware being used for the computations.

Therefore, we use the number of states visited by the player in playing the game as the measure of computational resources. We are careful to measure this across the multiple samples of an information set and to include the states visited in the backtracking of invalid samples. So we measure the states visited in creating the samples and the states visited using the samples as starting points for playouts.

7. samples of an information set
8. 4 x 2 x 10 x 4400, remember the Imperfect-Information Simulation starts at the beginning of the game, not the next round

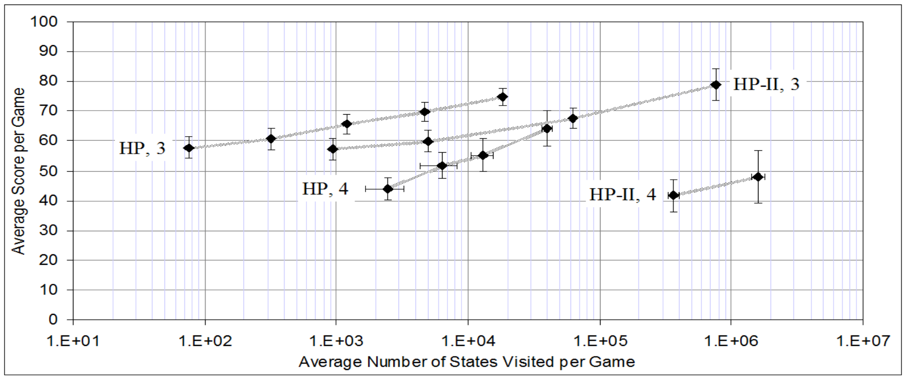

Figure 9: Hidden Connect game showing the HP player and the HyperPlay-II player performance

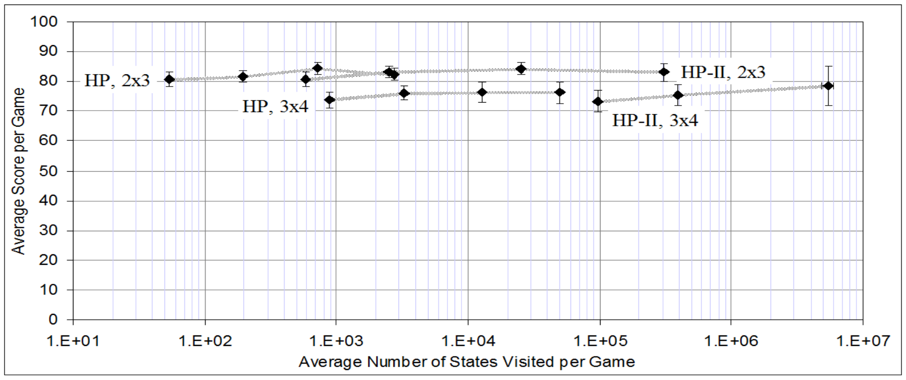

Figure 10: Mastermind game showing the HP player and the HyperPlay-II player performance

H.4 Hidden Connect

We used two variants of the game, connect 3 in a 3x3 grid and connect 4 in a 5x5 grid. As this game does not have any information-gathering or information-purchasing moves the HP player performed as well as the HyperPlay-II player but consumed considerable fewer resources with both players improving their performance as they visited more states.

H.5 Mastermind

We used two variants of the game, two colors in three positions with three guesses, and three colors in four positions with four guesses. The reward was pro rata for the number of correct positions. This game has no hidden game play, only a hidden initial setting by the random player. The HyperPlay-II player consumed considerable more resources for no additional improvement in performance.

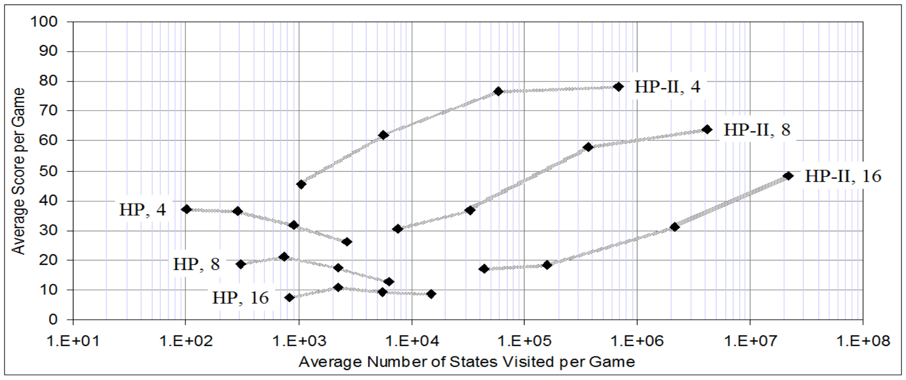

Figure 11: Number Guessing game showing the HP player and the HyperPlay-II player performance

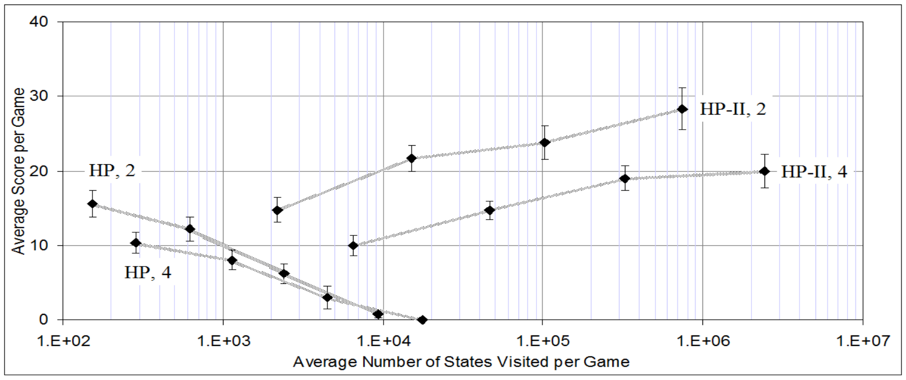

Figure 12: Banker and Thief game showing the HP player and the HyperPlay-II player performance

H.6 Number Guessing

We use variants of 4, 8 & 16 numbers. As expected, the HP player was unable to correctly value the information gathering moves and performed no better than a random player would. Whereas, the HyperPlay-II player tended towards optimum play as the resources were increased.

H.7 Banker and Thief

We use variants with 2 and 4 banks and a deposits of 10 by \$10.00. As expected the HP banker uses a greed strategy when making deposits and falls victim to the thief. Whereas, the HyperPlay-II player tended towards optimum play as the resources were increased[9].

9. optimum play rewards \$40.00 by creating a false target of \$60.00

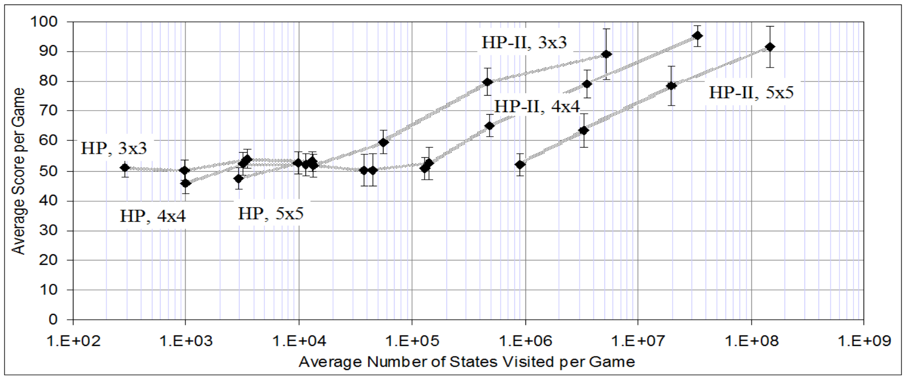

Figure 13: Battleships in Fog game showing the HP player and the HyperPlay-II player performance

H.8 Battleships in Fog

We use 3x3, 4x4 and 5x5 grid variants with a game length of 10 moves. This is a tactical game where playes must evaluate every round for a tactical advantage. The HP player plays little better than random with a score just above 50, due to some lucky first shots. Whereas, the HyperPlay-II player tends towards optimum play as the resources are increased.

References

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

Love, N., Hinrichs, T., Schkufza, D. H. E., & Genesereth, M. (2006). General game playing: Game description language specification. Tech. rep. LG–2006–01, Stanford Logic Group.

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

Schofield, M., & Thielscher, M. (2015). Lifting HyperPlay for general game playing to incomplete-information models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 3585–3591.

Schofield, M., & Thielscher, M. (2016). The scalability of the HyperPlay technique for imperfect-information games. In *Proceedings of the AAAI Workshop on Computer Poker and Imperfect Information Games*.

Schofield, M., & Thielscher, M. (2017). The efficiency of the HyperPlay technique over random sampling. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 282–290.

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