Lifting Model Sampling for General Game Playing to Incomplete-Information Models

Michael Schofield and **Michael Thielscher**
School of Computer Science and Engineering
The University of New South Wales
{mschofield, mit}@cse.unsw.edu.au

Abstract

General Game Playing is the design of AI systems able to understand the rules of new games and to use such descriptions to play those games effectively. Games with incomplete information have recently been added as a new challenge for general game-playing systems. The only published solutions to this challenge are based on sampling complete information models. In doing so they ground all of the unknown information, thereby making information gathering moves of no value; a well-known criticism of such sampling based systems. We present and analyse a method for escalating reasoning from complete information models to incomplete information models and show how this enables a general game player to correctly value information in incomplete information games. Experimental results demonstrate the success of this technique over standard model sampling.

Introduction

General Game Playing (GGP) is concerned with the design of AI systems able to take the rules of any game described in a formal language; and to play that game effectively.

Games with perfect information have made advances due, mostly, to the standardisation of the Game Description Language (GDL) (Love et al. 2006) and its widespread adoption. For example, in the AAAI GGP Competition (Genesereth, Love, and Pell 2005). Successful players employ either: automatically generated evaluation functions (Clune 2007; Schiffel and Thielscher 2007); or some form of Monte Carlo technique such as the modern UCT (Björnsson and Finnsson 2009; Méhat and Cazenave 2011).

Games with *imperfect*[1] *information* have recently been added as a new challenge for existing general game playing systems (Thielscher 2010). However little progress has been made on these types of games beyond a specification of their rules (Quenault and Cazenave 2007; Thielscher 2011). The only published approaches to modelling and playing incomplete information games (Edelkamp, Federholzner, and Kissmann 2012; Schofield, Cerexhe, and Thielscher 2012) both use standard model sampling as a partial solution to this challenge. The sampled states are then used to form the root of a separate *perfect information* search.

An obvious criticism of these sampling methods is that they value information at zero because each individual sample assumes perfect information at all times (Frank and Basin 2001). This was confirmed at the incomplete information track of the GGP competition at the Australasian Joint Conference on Artificial Intelligence; where three games, NumberGuessing, BankerAndThief, BattleshipsInFog, were specifically designed to test the ability of players to value information. None of the competitors were able to do so.[2]

In this paper we propose HyperPlay-II (for: HyperPlay with incomplete information models) as an extension of the original model sampling technique HyperPlay (Schofield, Cerexhe, and Thielscher 2012). This extended technique is able to play a much larger class of games by reasoning on incomplete information models. Our analysis and experimental results show that the new technique values information correctly according to the expected cost/benefit, performs information gathering moves when appropriate, is protective of information that should remain discreet, and requires no additional resources over its predecessor.

This paper is organised as follows. First, we recapitulate the basics of General Game Playing and related work. We formally present both techniques[3]; then we report on experimental comparisons using the aforementioned games. We conclude with a short discussion.

Related Work

The declarative game description language (GDL) has been defined as a formal language that allows an arbitrary game to be specified by a complete set of rules (Genesereth, Love, and Pell 2005). It uses a logic programming-like syntax and is characterised by a few keywords. These are highlighted in an example shown below, in Figure 1.

Originally for complete information games, GDL has been extended to GDL-II (for: *GDL with imperfect information*) to allow for arbitrary, finite games with randomised moves and incomplete information (Thielscher 2010).

The published approaches to designing general game players for GDL-II show how existing complete information

[1]GGP uses "imperfect" where general AI uses "incomplete". We will use "incomplete".

[2]see https://wiki.cse.unsw.edu.au/ai2012/GGP.

[3]The new technique is built on the formalism of the old.

```
1 role(agent).
2 role(random).
3
4 colour(red).
5 colour(blue).
6
7 init(round(0)).
8
9 legal(random,arm(C)) :- colour(C), true(round(0)).
10 legal(random,noop)   :- not true(round(0)).
11 legal(agent,noop)    :- true(round(0)).
12 legal(agent,ask)     :- true(round(1)).
13 legal(agent,wait)    :- true(round(1)).
14 legal(agent,cut(C))  :- true(round(2)), colour(C).
15
16 sees(agent,C) :- does(agent,ask), true(armed(C)).
17 next(round(1))    :- true(round(0)).
18 next(round(2))    :- true(round(1)).
19 next(round(3))    :- true(round(2)).
20 next(armed(C))    :- does(random,arm(C)).
21 next(armed(C))    :- true(armed(C)).
22 next(score(90))   :- does(agent,ask).
23 next(score(100))  :- does(agent,wait).
24 next(score(S))    :- true(score(S)), not explodes.
25 next(score(0))    :- explodes.
26
27 explodes :- does(agent,cut(C)), not true(armed(C)).
28
29 terminal :- true(round(3)).
30 goal(agent, S) :- terminal, true(score(S)).
31 goal(agent, 0) :- not terminal.
32 goal(random,0).
```

Figure 1: GDL-II description of the `ExplodingBomb` game.

GDL players can be lifted to play general incomplete information games by using models as the starting states for a complete information search (Edelkamp, Federholzner, and Kissmann 2012; Schofield, Cerexhe, and Thielscher 2012). This has been motivated by set sampling (Richards and Amir 2009) and by particle system techniques (Silver and Veness 2010). Similar special case applications of sampling to reduce incomplete to complete information can be found in (Ginsberg 2001; Kupferschmid and Helmert 2007).

Beyond GGP, Frank and Basin (2001) have investigated incomplete information games with a focus on Bridge, presenting a 'game-general' tree search algorithm that exploits a number of incomplete information heuristics. The Alberta Computer Poker Research Group has developed systems at the forefront of computer Poker players (Billings et al. 2006). This is a challenging domain combining incomplete and misleading information, opponent modelling, and a large state space. While not explicitly interested in GGP, they do describe several techniques that could generalise to this field, including *miximix*, fast opponent modelling, and Nash equilibrium solutions over an abstracted state space.

Example

The GDL-II rules in Figure. 1 formalise a simple game that 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.

The intuition behind the GDL rules is as follows.[4] Lines 1-2 introduce the players' names. Line 7 defines the single feature that holds in the initial game state. The possible moves are specified by the rules for keyword `legal`: in the first round, `random` arms the bomb (line 9), then the agent gets to choose whether to ask or wait (lines 12-13), followed by cutting a wire of their choice (line 14). The agent's only percept is the true answer if they enquire about the right wire (line 16). The remaining rules specify the state update (rules for `next`), the conditions for the game to end (rule for `terminal`), and the payoff (rules for `goal`).

[4]A word on the syntax: We use a Prolog-style notation for GDL rules as we find this more readable than the usual prefix notation used at competitions. Variables are denoted by uppercase letters.

Lifting HyperPlay

Sampling the information set grounds all unknown information. The move selection calculations based on such a grounding will value information gathering moves at zero; as all the information has already been gathered. This is HyperPlay's Achilles' heel and it motivates this work.

For example, the standard model sampling technique is unable to correctly play the ExplodingBomb game in Figure 1 because it incorrectly extrapolates individual samples of the information set to fact rather than treating multiple samples in concert: One sample 'knows' the armed wire is red, another 'knows' it is blue. Each sample then cuts the 'right' wire without acknowledging that it may be wrong.

To remedy this weakness we present an extended and refined technique that includes an Incomplete Information Simulation (IIS) in the decision making process. The IIS reasons directly with incomplete information, exploring the consequences of every action in its context, and uses the outcomes to make a decision. The new technique encompasses larger (i.e. non-singleton) subsets of the information set. The result is that it places the correct value on knowledge and will choose information gathering moves and information protecting moves when appropriate.

Formalism for HyperPlay-II

We proceed by formally describing the HyperPlay-II process. As with the original approach, the new technique requires a bag of models of the information set, representing a weighted sample. This is similar to a weighted particle filter in that all unknowns are grounded, and each model is updated based on moves (actions) and percepts (signals).

We adopt notation for finite games in extensive form; we refer to the original technique (Schofield, Cerexhe, and

Thielscher 2012) and we refer to (Thielscher 2011) for definitions that describe formally how any GDL-II description in logic defines a finite game tree, and how joint moves (tuples) and information sets are determined by the percepts received by each role. The rigorously formal exposition is to demonstrate the correctness of the technique and to provide sufficient detail to enable others to replicate the method.

GDL Game

Definition 1. Let $G = \langle S, R, A, \Sigma, v, do, sees \rangle$ be an incomplete information game given by a GDL-II description:

- S is a set of states, or nodes on the game tree;
- R is a set of roles in the game;
- A is a set of moves in the game, and $A(s, r) \subseteq A$ is a set of legal moves, for role $r \in R$ in state $s \in S$;
- Σ is a set of percepts in the game, and $\sigma \in \Sigma$ is a percept[5], given by the $sees()$ function below;
- $v : S \times R \to \mathbb{R}$ is the payoff function on termination;
- $do : S \times A^{|R|} \to S$ is the successor function; and
- $sees : S \times A^{|R|} \to \Sigma^{|R|}$ is the percept function.

Additionally we use the following notation:

- $s_0 \in S$ for the initial state and $s_t \in S$ for the true state;
- T as the set of the terminal states;
- $D = S \backslash T$ as the set of decision states;
- $\mathcal{H}$ is the information partition of D;
- $H_r \in \mathcal{H}$ is the information set for $r \in R$;
- $h_r \in H_r$ is a state in the information set for $r \in R$; and
- $\phi(A)$ is a probability distribution across set A.

We have defined a game with states, roles, moves, percepts, a successor function, and a payoff on termination; now we must define the move vector and percept vector that facilitates the succession of the game.

Definition 2. Let G be a GDL-II game defined above, then:

- $a_r \in A(d, r)$ is a move for role $r \in R$ in state $d \in D$;
- $\vec{a} = \langle a_1 ... a_{|R|} \rangle$ is a move vector, one move for each role;
- $\langle \vec{a}_{-r}, a_r \rangle = \langle a_1 \dots a_r \dots a_{|R|} \rangle$ is a move vector containing a specific move a_r for role $r \in R$;
- $\sigma \in \Sigma$ given by $sees(d, \vec{a})$ is a percept for actions $\vec{a}$ in state $d \in D$;
- $\vec{\sigma} = \langle \sigma_1 ... \sigma_{|R|} \rangle$ is a percepts vector; and
- $s = do(d, \vec{a})$ and $\vec{\sigma} = sees(d, \vec{a})$ is the natural progression of the game.

To illustrate we look at the Exploding Bomb game shown in the GDL of Figure 1. The successor function that progresses the game from state h_0 in the information set shown in Figure 2 is: $s = do(h_0, \langle ask, noop \rangle)$ and the percepts arising from those actions are $\langle red, null \rangle = sees(h_0, \langle ask, noop \rangle)$.

[5] Multpile percepts are considered to be conjoined into one. "null" is the empty percept.

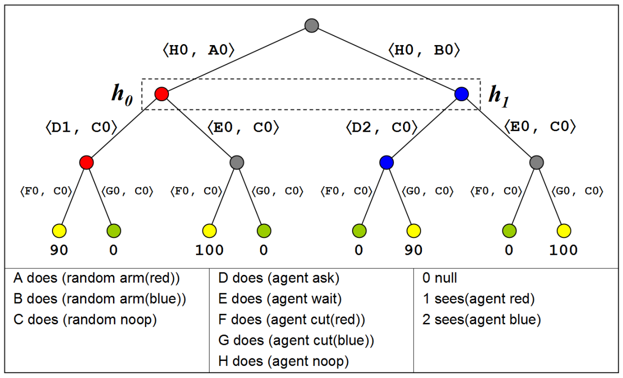

Figure 2: The game tree for the Exploding Bomb. At Round 1 the agent has an Information Set of two states; that is, $H_{agent} = \{h_0, h_1\}$.

Move Selection Policy

In order for a game to be played out to termination we require a move selection policy π_r for each role, being an element of the set of all move selection policies Π.

Definition 3. Let G be a GDL-II game defined above, then:

- $\Pi : D \times R \to \phi(A)$ is a move selection policy expressed as a probability distribution across A;
- $\vec{\pi} : \langle \pi_1, .., \pi_{|R|} \rangle$ is a tuple of move selection policies; and
- $play : S \times \Pi^{|R|} \to \phi(T)$ is the playout of a game to termination according to the given move selection policies.

In our example, the playout of the game from state h_0 to termination by two Monte Carlo players would be $play(h_0, \vec{mc})$ with $\phi(T) = \langle 0.25, 0.25, 0.25, 0.25 \rangle$.

Move Evaluation and Selection

Move selection requires an evaluation function $eval()$. We play out the game according to a move selection policy and use the terminal value as a measure of utility.

Definition 4. Let G be a GDL-II game defined above, then:

- $\phi(s)$ is the *a priori* probability that $s = s_t$;[6]
- $eval : S \times \Pi^{|R|} \times R \times \mathbb{N} \to \mathbb{R}$ is the evaluation function; where
- $eval(s, \vec{\pi}, r, n) = \frac{1}{n} \sum_1^n \phi(s) \times v(play(s, \vec{\pi}), r)$ evaluates the node $s \in S$ using the policies in $\vec{\pi}$, and sample size n.

Applying the evaluation function to move vector $\langle \vec{a}_{-r}, a_{ri} \rangle$, we get:

- $eval(do(d, \langle \vec{a}_{-r}, a_{ri} \rangle), \vec{\pi}, r, n)$ is the evaluation; and
- $argmax_{a_{ri}}[eval(do(d, \langle \vec{a}_{-r}, a_{ri} \rangle), \vec{\pi}, r, n)]$ is a selection process for making a move choice.

The agent has an opportunity (in round 1) to make the desired move $\langle ask \rangle$. The expected value of a Monte Carlo playout is: $eval(do(h_0, \langle \vec{a}_{-r}, ask \rangle), \vec{mc}, agent, 4) = 45$; whereas the expected value of a Monte Carlo playout for the move $\langle wait \rangle$ is 50.

[6] State s may be uncertain; eg. $\sum_{h \in H} \phi(h) = 1$ for the information set H.

State Identification

Each state in the game tree is uniquely identified by an ordered list of play messages (a path) representing the moves and percepts of *all* the players in the game, starting from an initial state and ending at the state in question.

Definition 5. Let G be a GDL-II game defined above, then:

- $p \in P$ is a play message from the set of all play messages in game G;
- $p_r = \langle a_r, \sigma_r \rangle$ is a play message for role $r \in R$;
- $\vec{p} = \langle p_1, ...p_{|R|} \rangle$ is a play message vector;
- $\xi : S \to P^*$ is a function that extracts the path of a state;
- $\xi(s) = \langle \vec{p}_0, ...\vec{p}_n \rangle$ is an ordered list of play messages from initial state s_0 that defines state s called a path; and
- $\xi_r(s) = \langle p_{r0}, ...p_{rn} \rangle$ is an ordered list of play messages received by role $r \in R$ called an **incomplete** path.

It is the fact that the path is incomplete that defines the players information set H_r. If the path is complete then the information set collapses to the true state; ie. $H_r = \{s_t\}$.

The right most path in Figure 2 is defined by the play messages on the left. However the agent receives an incomplete path (right), so the true path may be ambiguous.

$$\begin{array}{ll} \langle H0, B0 \rangle & \langle H0, \quad \rangle \\ \langle E0, C0 \rangle & \langle E0, \quad \rangle \\ \langle G0, C0 \rangle & \langle G0, \quad \rangle \end{array}$$

The HyperPlay Technique

We see that the HyperPlay technique (Schofield, Cerexhe, and Thielscher 2012) constructs an information set from an incomplete path by grounding the unknown values, then selects a move that maximises expected utility.

Definition 6. Let G be a GDL-II game defined above, then:

- $hp : P^n \to 2^{P^n}$ is the function HyperPlay that completes a path in game G to sample the information set H; and
- $h_r \in H_r = hp(\xi_r(s_t))$ where s_t is the true game state.

The Hyperplay function completes the incomplete path consistent with the game description. Note the new, legal path (right) is different to the true path shown previously.

$$\begin{array}{ll} \langle H0, \quad \rangle & \langle H0, A0 \rangle \\ \langle E0, \quad \rangle & \langle E0, C0 \rangle \\ \langle G0, \quad \rangle & \langle G0, C0 \rangle \end{array}$$

Definition 7. Let G be a GDL-II game defined above, then:

- $argmax_{a_{ri}} [\sum_{j=1}^{|H_r|} eval(do(h_{rj}, \langle \vec{a}_{-r}, a_{ri} \rangle), \vec{mc}, r, n)]$
  is the move selection policy π_{hp}, where $\vec{mc}$ is random.

In our example, reasoning on perfect information with a Monte Carlo move selection policy gives the following,

$$\begin{aligned} & eval(do(h_i, ask), \vec{mc}, agent, 4) \\ & \quad = 0.25 \times (90 + 0 + 90 + 0) = 45 \\ & eval(do(h_i, wait), \vec{mc}, agent, 4) \\ & \quad = 0.25 \times (100 + 0 + 100 + 0) = 50 \quad (selected) \end{aligned}$$

Incomplete Information Simulation

For the correct valuation of information gathering moves, the player must be able to conduct a playout with incomplete information, as opposed to a Monte Carlo playout.

To do this we take the path for a state in the information set and use it for an IIS with a HyperPlayer in each role. It should be noted that this requires a playout from the initial state, not just the state in the information set.

Definition 8. Let G be a GDL-II game defined above, then:

- $replay : S \times P^n \times \Pi \to S^{|R|}$ is the replay of a game consistent with the path of a state;
- $replay(s_0, \xi_i(h_r), \vec{\pi})$ is the replay of a game, as if $h_r = s_t$ and generating information sets for all roles, such that, $hp(\xi_i(h_r)) \to H_i$ where r is our role and i is any role;
- $IIS : S \times \Pi \times R \times \mathbb{N} \to \mathbb{R}$ is the incomplete information simulation; where
- $IIS(h_r, \vec{\pi}_{hp}, r, n)$ is an evaluation using an incomplete information simulation, and is defined as $eval(replay(s_0, \xi_i(h_r), \vec{\pi}_{hp}), \vec{\pi}_{hp}, r, n)$.

The IIS generates multiple paths. In row one we see what the agent knows; row two is a model in the agent's information set; row three shows the two incomplete paths created by the IIS; and row four shows models from the information sets of each of the IIS roles, which may represent a state outside the agent's information set.

1	$\xi_a(s_t)$		$\langle H0, \quad \rangle$ $\langle E0, \quad \rangle$	
2	$hp(\xi_a(s)) \to h_a$		$\langle H0, A0 \rangle$ $\langle E0, C0 \rangle$	
3	$\xi_i(h_a)$	$\langle H0, \quad \rangle$ $\langle E0, \quad \rangle$		$\langle \quad , A0 \rangle$ $\langle \quad , C0 \rangle$
4	$hp(\xi_i(h_a))$	$\langle H0, B0 \rangle$ $\langle E0, C0 \rangle$		$\langle H0, A0 \rangle$ $\langle D0, C0 \rangle$

The HyperPlay-II Technique

We now show a formalism for the HyperPlay-II technique.

Definition 9. Let G be a GDL-II game defined above, then:

- $a_{ri} \in A(h_r, r)$ is a move to be evaluated by role r;
- $\langle \vec{a}_{-r}, a_{ri} \rangle$, is the move vector containing the move a_{ri};
- $IIS(do(h_r, \langle \vec{a}_{-r}, a_{ri} \rangle), \vec{\pi}_{hp}, r, n)$ is an evaluation; and
- $argmax_{a_{ri}} [\sum_{j=1}^{|H_r|} IIS(do(h_{rj}, \langle \vec{a}_{-r}, a_{ri} \rangle), \vec{\pi}_{hp}, r, n)]$
  is the move selection policy $\vec{\pi}_{hpii}$.

Reasoning on incomplete information using the move selection policy $\vec{\pi}_{hp}$ gives the following,

$$\begin{aligned} & IIS(do(h_i, ask), \vec{\pi}_{hp}, agent, 4) \\ & \quad = 0.25 \times (90 + 90 + 90 + 90) = 90 \quad (selected) \\ & IIS(do(h_i, wait), \vec{\pi}_{hp}, agent, 4) \\ & \quad = 0.25 \times (100 + 0 + 100 + 0) = 50 \end{aligned}$$

Note the similarity with the move selection policy $\vec{\pi}_{hp}$ given in Definition 7. In this respect we characterise the new technique as a nested player.

The use of the IIS extends the domain of reasoning to the least upper bound of the information partition $sup\mathcal{H} \subseteq D$. As the $hp()$ function generates paths across an information domain that is the closed with respect to what the other roles can know, which arises from the information set of our role:

- $hp(\xi_r(s_t))$ generates the Information Set H_r; and
- $hp(\xi_i(h_r))$ for $h_r \in H_r$ generates an Information Set H_i for any role in the IIS.

It is both the expanded domain and the use of incomplete information reasoning that gives the new technique an advantage over its predecessor. That is to say, there is an improvement in both the quantitative and qualitative aspects of the player.

There is, however, a class of games that it cannot play. We discuss this class of games in the limitations section, after the experimental results.

Experiments

A series of experiments was designed to test the capabilities of the new technique using the high-throughput computer facilities at the School of Computer Science and Engineering.

Games played at the recent Australasian Joint Conference on Artificial Intelligence 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 organisers specially designed games that would challenge the state of the art of GDL-II players so as to encourage research and development in this field.

As with previous research, we have modelled the game server and both players in a single thread so it could be parallelised across many CPUs. Each test was repeated one hundred times. Error bars indicate a 99% confidence interval.

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.

The new technique has the potential to be a 'resource pig' as it is a nested player. For example, a value of $n = 4$ in $eval(s, \vec{\pi}, r, n)$ for the original technique in a two player game with a branching factor of eight and a playout length of 10 would result in 1,280 states being visited. The new technique would visit 2,048,000 states.

However experiments showed that the new technique can function optimally[7] with the same *total* resources as the original player. That is, where the original player might need $n = 16$, the new player only requires $n = 4$. This translates to the same number of states visited by each player.

[7]A player is 'optimal' when increasing its resources will not lead to better play. We refer to this as 'adequate resourcing'.

Equal Resources

Some experiments were conducted with two-player games, pitting the original player against the new player using equal resources. A resource index $n = 4$ gives the new player four hypergames, each running an IIS with four hypergames. The old player would get $n = 16$ hypergames.

A player resource index of zero represents random decision making and serves to provide a basis for improvement.

Results

The results of each experiment are given below, along with a brief commentary on their significance.

Exploding Bomb

The old 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. Table 1 shows the results of calculations made by each technique when choosing an action. The action with the highest expected score is chosen for each round (shown in bold). The original technique chooses randomly in round 2.

round	agent does	HyperPlay	HyperPlay-II
1	ask	45.04 ± 0.09	$\mathbf{90.00 \pm 0.00}$
1	wait	$\mathbf{49.98 \pm 0.10}$	49.91 ± 0.64
2	cut unarmed	$\mathbf{49.40 \pm 1.19}$	0.00 ± 0.00
2	cut armed	$\mathbf{50.60 \pm 1.19}$	$\mathbf{90.00 \pm 0.00}$

Table 1: Experimental score calculations during the Exploding Bomb decision making process

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.

Table 2 shows experimental results in the form of calculated expected outcomes (the chosen action is bold). The original HyperPlayer always tells to avoid the penalty. HyperPlayer-II recognises that hiding this information yields a better expected outcome.

arming agent does	HyperPlay	HyperPlay-II
arm blue and tell	$\mathbf{60.00 \pm 0.15}$	20.00 ± 0.00
arm red and tell	$\mathbf{60.04 \pm 0.14}$	20.00 ± 0.00
arm blue and hide	39.98 ± 0.16	$\mathbf{40.36 \pm 1.22}$
arm red and hide	39.99 ± 0.14	$\mathbf{39.45 \pm 1.33}$

Table 2: Expected score calculations for the arming agent in round one of the Spy vs. Spy decision making process

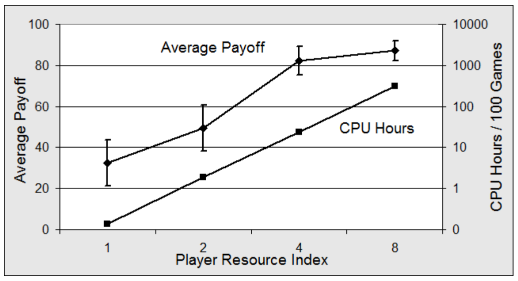

Figure 3: The NumberGuessing Results for HyperPlay-II.

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.

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 3 the new player approaches this score.

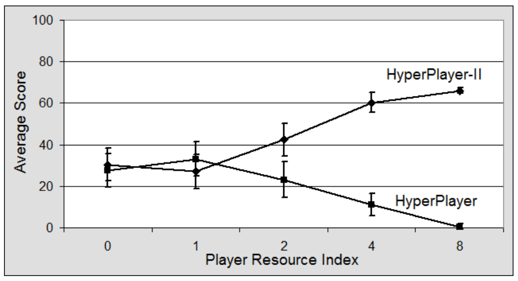

Figure 4: The Banker and Thief results.

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.

Figure 4 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 bank.

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

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.

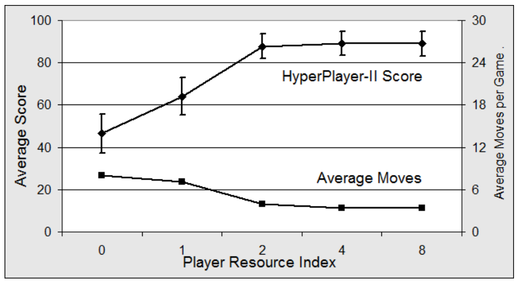

Figure 5: The Battleships In Fog results.

The original player sees no value in scanning as all of the samples ‘know’ where the opponent is. It doesn’t 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). This is reflected in Figure 5. Note also that HyperPlayer-II requires only three rounds to finish the game: scan, noop, fire.

Conclusion

The experimental results show the value HyperPlay-II places on knowledge, and how it correctly values information-gathering moves by it and its opponents. It is able to collect information when appropriate, withhold information from its opponents, and keep its goals secret. The use of the Incomplete Information Simulations is an efficacious and efficient tool for reasoning with incomplete information. A HyperPlayer-II was easily able to outperform an equally resourced HyperPlayer in all of the experiments.

We intend to explore additional features of the HyperPlay-II technique as they pertain to general artificial intelligence. We also intend to implement the HyperPlayer-II for the General Game Playing arena as a benchmark competitor for other researchers to challenge in GDL-II games.

Limitations

There is an interesting type of games requiring what is known as coordination without communication (Fenster, Kraus, and Rosenschein 1995) that goes beyond what our technique can achieve.

Consider the following *cooperative* variant of the Spy vs. Spy game. Spy1 sees which wire is used to arm a bomb. They then signal the name of a colour to Spy2, who must try to disarm the bomb. *Both* win if Spy2 cuts the right wire and lose otherwise. Clearly Spy1 has an incentive to help Spy2, and there is one obvious way to do this: signal the colour of the wire. The crux, however, is that the game rules can be designed such that the colour being signalled is logically independent of the colour of the armed wire.

Acknowledgments

This research was supported by the Australian Research Council under grant no. DP120102023. The second author is also affiliated with the University of Western Sydney.

References

Billings, D.; Davidson, A.; Schauenberg, T.; Burch, N.; Bowling, M.; Holte, R.; Schaeffer, J.; and Szafron, D. 2006. Game-tree search with adaptation in stochastic imperfect-information games. In *Proc. Computers and Games*, 21–34.

Björnsson, Y., and Finnsson, H. 2009. CadiaPlayer: A simulation-based general game player. *IEEE Transactions on Computational Intelligence and AI in Games* 1:4–15.

Clune, J. 2007. Heuristic evaluation functions for general game playing. In *Proc. AAAI*, 1134–1139.

Edelkamp, S.; Federholzner, T.; and Kissmann, P. 2012. Searching with partial belief states in general games with incomplete information. In *Proc. KI*, 25–36.

Fenster, M.; Kraus, S.; and Rosenschein, J. S. 1995. Coordination without communication: Experimental validation of focal point techniques. In *ICMAS*, 102–108.

Frank, I., and Basin, D. 2001. A theoretical and empirical investigation of search in imperfect information games. *Theoretical Computer Science* 252(1-2):217–256.

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

Ginsberg, M. L. 2001. GIB: Imperfect information in a computationally challenging game. *J. Artif. Intell. Res.(JAIR)* 14:303–358.

Kupferschmid, S., and Helmert, M. 2007. A Skat player based on Monte-Carlo simulation. In *Proc. Computers and Games*, 135–147.

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

Méhat, J., and Cazenave, T. 2011. A parallel general game player. *KI-Künstliche Intelligenz* 43–47.

Quenault, M., and Cazenave, T. 2007. Extended general gaming model. In *Computer Games Workshop*, 195–204.

Richards, M., and Amir, E. 2009. Information set sampling for general imperfect information positional games. In *Proc. IJCAI Workshop on GGP*, 59–66.

Schiffel, S., and Thielscher, M. 2007. Fluxplayer: A successful general game player. In *Proc. AAAI*, 1191–1196.

Schofield, M.; Cerexhe, T.; and Thielscher, M. 2012. HyperPlay: A solution to general game playing with imperfect information. In *Proc. AAAI*, 1606–1612.

Silver, D., and Veness, J. 2010. Monte-Carlo planning in large POMDPs. In *Proc. NIPS*, 2164–2172.

Thielscher, M. 2010. A general game description language for incomplete information games. In *Proc. AAAI*, 994–999.

Thielscher, M. 2011. The general game playing description language is universal. In *Proc. IJCAI*, 1107–1112.