The Workshops of the Thirtieth AAAI Conference on Artificial Intelligence
Computer Poker and Imperfect Information Games: Technical Report WS-16-06

The Scalability of the HyperPlay Technique for Imperfect-Information Games

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

Abstract

In the field of General Game Playing the imperfect-information games present a special challenge for researchers. In general the search space is larger, and the lack of information requires a different decision making technique. A simple Monte Carlo sampling using a particle filter may serve for the simple games, but this soon fails when more complex games are played. The HyperPlay technique was one such "simple" player, soon enhanced to HyperPlay-II[1] capable of handling the most complex of games. However, this technique is very resource hungry and may not scale well for larger games. We explore the scalability of HyperPlay-II for a variety of imperfect-information games and test some perfect-information pruning techniques to see if they will improve efficiency.

Introduction

General Game Playing is a branch of AI research focused on agents capable of playing any game with no prior experience (Genesereth, Love, and Pell 2005). The agent is given a set of declarative rules[2] and is expected to play the game as well as any human player might.

Within the General Game Playing (GGP) domain the most popular games are those with perfect information as to the reward structure and the game play (Genesereth and Björnsson 2013). Games with imperfect information present a special challenge for AI players as the game play is hidden and replaced with percepts. So a player may only receive limited information about the state of play, and hence the state of the game.

Incomplete-Information Games

In the GGP community the term "imperfect" is used where general AI uses the term "incomplete." We will use "imperfect" information to mean that the game play may be hidden from one or more players. The initial state of the game and reward structure is known as they are declared in the rules.

These games have been added as a challenge for existing GGP systems (Thielscher 2010; Schiffel and Thielscher 2014), presenting several unique challenges for the AI player. Firstly, the search space is often larger than similar perfect-information games as the player must search parts of the game tree that would otherwise be known to be inaccessible. Secondly, the player must reason across an information set[3] and choose the move that is most likely to give a positive outcome.

HyperPlay Technique

We recapitulate a brief description of the HyperPlay (HP) and HyperPlay-II (HP-II) techniques (Schofield, Cerexhe, and Thielscher 2012; Schofield and Thielscher 2015).

HyperPlay is a bolt on solution for a perfect-information player that allows it to play imperfect-information games. There is no limitation to the type of perfect-information player, however in this paper we choose to use a simple player based on Monte Carlo sampling.

The HP technique converts the perfect-information player by maintaining a bag of models of the true game, updating them as new information[4] comes to hand. The HP technique uses the bag of models to create a sample of the information set, as well as a probability of the model being the true game. In effect we have a weighted particle filter.

Using the weighted particle filter the HP player can conduct a perfect-information search and combine the expected outcomes using the appropriate weighting. This process is not well behaved as GDL-II allows for players to make illegal moves in the true game. Thus, a move that is legal in the sample may not be legal in the true game. So the sample weightings also express the likelihood that the moves are legal in the true game.

The obvious flaw is the elevation of sample to fact via the perfect-information search, which is overcome in the HP-II technique by the use of nested playouts or Imperfect-Information Simulations. This has proved very successful in correctly valuing information gathering moves, but also very costly in terms of computational resources.

There is one important difference between the Imperfect-Information Simulations and a conventional playout. That is, a conventional playout starts from a sample of the in-

[1] for Imperfect Information

[2] an example is given in Figure 1

[3] that is, the set of possible game states that satisfies all the percepts received by the player

[4] percepts received from the game controller in the GDL form `(sees player percept)`

```
(role player)     (role random)
(init (step 0))   (init starttime)
(number 1)  ...   (number 16)
(<= (legal random (choosenumber ?n))
    (number ?n) (true starttime))
(<= (legal player (ask lessthan ?n))
    (number ?n) (true questiontime))
(<= (legal player readytoguess) (true questiontime))
(<= (legal player (guess ?n))
    (number ?n) (true guesstime))
...
(<= (sees player yes)
    (does player (ask (lessthan ?n)))
    (true (secretnumber ?m)) (less ?m ?n))
(<= (next (secretnumber ?n))
    (does random (choosenumber ?n)))
(<= (next questiontime) (true starttime))
(<= (next guesstime) (does player readytoguess))
(<= (next right)
    (does player (guess ?n)) (true (secretnumber ?n)))
(<= (next wrong)
    (does player (guess ?n)) (not (true secretnumber ?n)))
...
(<= terminal (true right))
(<= terminal (true wrong))
(<= (goal player 100) (true right) (true (step 3)))
(<= (goal player  90) (true right) (true (step 4)))
... (<= (goal player 0) (true wrong))
```

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

formation set and plays to termination, but the Imperfect-Information Simulations starts from the initial state of the game and passes through the sample of the information set on its way to termination, agreeing with all the percepts received from the true game. This additional length of the playout adds to the resource requirement.

Related Research

The GGP community has been slow to pick up the challenge and only a few players have been implemented with even fewer competitions being conducted. The published approaches to designing general game players for GDL-II show how existing complete information GDL players can be lifted to play general imperfect-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). Edelkamp, Federholzner, and Kissmann (2012) have built a GDL-II player NEXUSBAUM based on perfect-information Monte Carlo sampling, which they compared against FLUXII, an imperfect-information extension of FLUXPLAYER (Schiffel and Thielscher 2007) that maintains the complete information set throughout a game.

Outside of GGP, Monte Carlo tree search has been applied to a variety of specific perfect- and imperfect-information games alike (Browne et al. 2012). Frank and Basin (1998) have presented a 'game-general' tree search algorithm for the card game Bridge that exploits a number of imperfect-information heuristics. For the same game, Ginsberg (2001) has applied perfect-information Monte Carlo sampling. Similar special case applications of sampling to reduce imperfect to perfect information can be found in Kupferschmid and Helmert (2007).

The so-called strategy fusion error has been identified as a main issue with straightforward perfect-information model sampling (Frank and Basin 1998). Long et al. (2010) analyse the impact of this error and present three conditions on game trees under which it does not have a strong adverse effect on the quality of play. For other games, the strategy fusion error has led to variations of perfect-information sampling that have been successfully applied to other card games (Wisser 2015). 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 imperfect and misleading information, opponent modeling, 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 modeling, and Nash equilibrium solutions over an abstracted state space.

Contribution

Our motivation for this research is the notion that the HP-II technique is a "resource pig" (Schofield and Thielscher 2015) as it uses nested playouts to evaluate move selections.

In this paper we focus on the HP-II technique and its consumption of computational resources for a particular level of performance. We make comparisons between HP and HP-II for a variety of games as well as measuring the increase in resources used by HP-II when a game is scaled up. We then implement several pruning techniques that have had some success in nested perfect-information players and measure the reduction in resources consumed.

This paper seeks to examine the cost of the imperfect-information aspects of a player, not the embedded perfect-information search techniques. While the latter is fertile ground for improvement, we focus on the resource consumed by the imperfect-information algorithms.

Game Description Language

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 Figure 1. GDL has been extended to GDL-II (for: *GDL with imperfect information*) to allow for arbitrary, finite games with randomised moves and imperfect information (Thielscher 2010).

Example Figure 1 shows some of the GDL-II rules for a simple game that commences with the *random* player selecting a number from 1 to 16, and then the player can ask yes/no questions about the number or announce that they are ready to guess. The intuition behind the GDL rules is as follows. Lines 1-2 introduce the roles' names and the initial game state, respectively. The possible moves are specified by the rules for keyword `legal`: in the first round, `random` chooses a number (lines 3-5), then the player can repeatedly enquire whether this number is less than another one (lines 6-7), until they declare their intention to guess (line 8) followed by picking their number (lines 9-10). The agent's only percept is the true answer to their questions (lines 12-14). The remaining rules specify the state update (rules for `next`), the conditions for the game to end (rules for `terminal`), and the payoff (rules for `goal`), which depends on the number of questions asked.

Imperfect-Information Games

In the General Game Playing domain for imperfect-information games, the rules of the game and the reward structure is fully known to each player. What is not automatically known are the moves made by other players in the game. Player receive percepts from the game controller according to the rules of the game expressed in the GDL-II. And so, we look at the variations that can occur in the structure of a game.

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.

There are many games that fit this category including board games that have been adapted for imperfect information. We use Hidden Connect to represent this type of game.

Hidden Connect Is a blind version of the children's two player board game, Connect 4.

Game Play is turn taking, with players dropping a coloured token into a vertical column of a grid. A player wins by making a row, column or diagonal of their tokens.

Reward Structure is constant sum with win, loss and draw possible for each player.

Player Percepts are received only when a column is full so that illegal moves are not made. Otherwise players get no clue as their opponents moves.

Optimal Play Strategy is a very basic modeling of expected outcomes. The only information gathering moves come indirectly by filling (or not filling) a column.

Game Tree is variable depth with a diminishing branching factor as columns fill up. The game tree contains around 10^{17} states for the five column version of the game.

Scalability is achieved by changing the number of columns and the number of tokens forming a line.

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.

There are many games that fit this category including many popular card games, We use the game Mastermind to represent this type of game.

Mastermind Is a guessing game where the random player selects a coloured pattern, and the player must replicate the pattern exactly.

Game Play is single player with the player offering coloured patterns for evaluation against a target.

Reward Structure is diminishing pro rata according to the correctness of the guess in a maximum number of turns.

Player Percepts are received every round about the closeness of the matching pattern, but are ambiguous.

Optimal Play Strategy is a to reduce the size of the information set as quickly as possible, and so a move that can halve the information set is an optimal move.

Game Tree is variable depth with a constant branching factor. The game tree contains around 10^9 states for the three colour version of the game.

Scalability is achieved by changing the number of colours and changing the number of tokens forming the pattern.

Information Purchasing

This type of game tests the players ability to value information gathering moves. Generally the player can choose between asking a question about the state of the game, or 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.

Number Guessing Is a guessing game where the random player selects a number from 1 to 16, and the player must ask questions about the number or guess the number directly.

Game Play is single player with the player asking questions like "is it less than x" or guess the number directly.

Reward Structure is diminishing per turn. As such, every question will cost the player a small portion of their final score.

Player Percepts are received every round in response to the question asked.

Optimal Play Strategy is to reduce the size of the information set as quickly as possible, and so a move that can halve the information set is an optimal move. A binary search is achievable in this game.

Game Tree is variable depth with a constant branching factor. The game tree contains around 10^{12} states for the 16 number version of the game.

Scalability is achieved by changing the number range.

Hidden Objectives

This type of game tests the players ability to identify their opponents 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.

Banker and Thief Is a strategy game where the random player gives the banker a secret target. The banker must avoid giving clues to the thief that will allow the thief to steal from the banker.

Game Play is a two player game with an uneven reward structure. The banker is given a target bank (from many banks) to make deposits, however if the thief can guess the target bank then the banker loses.

Reward Structure is uneven. The banker can keep all of the money deposited in the target bank after the thief has struck. The thief can only keep money stolen if it comes from the target bank.

Player Percepts are received every round about deposits made into each bank.

Optimal Play Strategy is to fool the thief into choosing the wrong target. As such the banker must give up more than half the money. A greedy strategy will always fail in this game.

Game Tree is fixed depth with a constant branching factor. The game tree contains around 10^7 states for the 4 bank, ten deposit version of the game.

Scalability is achieved by changing the number of banks and the number of deposits.

Tactical Information Valuing

This type of game tests 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.

Battleships in the Fog Is a tactical two player game with a random starting position.

Game Play is turn taking. Players can choose from three basic actions; scan, fire or move. Scanning reveals the location of their opponent, but tells them they have been scanned. Firing upon an adjacent square may lead to victory, but reveals the location of the square being fired upon. Moving changes a players location on the board.

Reward Structure is win/loss/draw. In practical terms the game is win/loss as a draw takes many moves of avoiding each other.

Player Percepts are received according to the action taken. Scanning gives the location of the scanned vessel to both players, and Firing gives the location of the square being fired upon.

Optimal Play Strategy is to correctly value the risks and rewards of moves that generate a percept. Ultimately this is a game of calculating what information your opponent knows about you based on what you know about your opponent. And hence, correctly valuing moves that reveal information.

Game Tree is variable depth with a constant branching factor. The game tree contains around 10^{33} states for the five-by-five grid.

Scalability is achieved by changing the size of the grid.

Heuristics and Pruning

Using a heuristic to improve the search and/or pruning the search space are effective ways to improve the computational efficiency of the move selection process for perfect information sampling. The extension of these techniques to the HP player seems straight forward, but there is no evidence to suggest that such techniques will work in the HP-II Imperfect-Information Simulation.

We examine several techniques implemented in a Nested Monte Carlo (NMC) player (Cazenave et al. 2016) as there is some similarity between the nesting examined in this paper and the nesting in the HP-II technique.

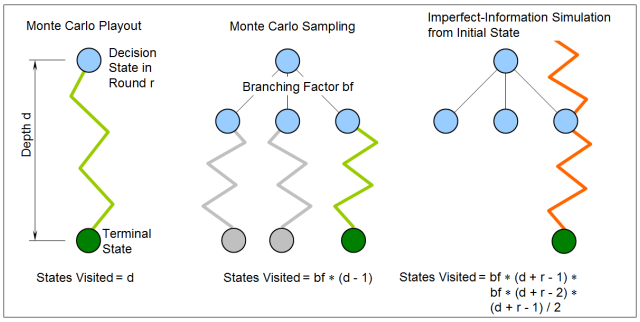

Figure 2: Playouts for a Monte Carlo Sampling and Nested Playouts for an Imperfect-Information Simulation

States Visited

The HP-II technique utilises a nested playout for evaluating move choices which causes a significant increase in the number of states visited during the analysis. As we are dealing with imperfect information the nested playout must start from the initial state of the game, not the current round. This doubles the number of states visited in a game compared to the perfect information version of a nested playout.

In Figure 2 we see the impact of the nested playout on the number of states visited. This explains the significant increase in computational resources required to play a game

of depth d; from $O(bf \cdot d^2)$ for the HP player to $O(bf^2 \cdot d^4)$ for the HP-II player.

Discounting

Discounting is a way of improving the information extracted from a playout to give a rich set of terminal results instead of the usual 0 or 100. Discounting based on the depth of the playout has been demonstrated (Cazenave et al. 2016) to improve the search performance and to facilitate search pruning.

A discount factor $0 < \gamma < 1$ is applied to the goal value to give the discounted value for decision making:

$v'(t, r) = v(t, r) \times \gamma^d$, where

$v(t, r)$ is the terminal value, and

d is the playout depth.

Discounting can only be effective in games with variable playout depth, so our player's performance will not be improved by discounting when playing the fixed depth games.

Pruning

We draw on two pruning techniques presented for NMC and adapt them to our player. In each technique we stop the playout when certain conditions are met. In each case the legal moves are evaluated in a random order so as not to bias the pruning process.

Cut on Win This technique works well with a NMC player in turn taking two player win/loss games, but has problems being implemented in games where players purchase information. The Cut on Win (CoW) technique requires a strict win/loss reward structure to be effective.

The player stops evaluating the legal moves when one move return the maximum possible score, that move is then selected. This can be problematic in games like Number Guessing as the maximum score is reduced each round. So we explore a variation where the player "knows" the maximum achievable score under optimal play conditions and uses that as a cut-off-point for the CoW pruning.

Pruning on Depth This technique also works well with a NMC player in turn taking two player win/loss games, but is ineffective when the playout depth is fixed.

The player makes decisions using discounted goal values and remembers the shortest depth to the maximum achievable score. All playouts are terminated when they exceed the shortest depth and a goal value of zero is returned. When multiple playouts are used for the evaluation, the player remembers the totals for both goal value and depth.

Implementation

We implement each of these techniques at both levels of the nested HP-II player, that is, the high level player and the embedded HP players in the Imperfect-Information Simulation, and report on the reduced computational resources and the change in performance level.

Design of Experiments

We design experiments to answer two basic questions:

- Does HP-II perform better than HP at this type of game, and at what computational cost; and
- What is the impact of up-sizing the game on the computational cost for HP-II to achieve the same level of performance.

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 so as to be competitive. Since we are making comparisons between different instantiations of our player, the experiments will not be overly sensitive to the performance of the opponent.

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 the information set and to include the states visited in the backtracking of invalid samples[5]. So we measure the states visited in creating the samples plus the states visited in the playouts.

Player Configuration

For the HP player we can alter two resource parameters: the size of the bag of models,[6] and the number of playouts per legal move. We say that HP(4, 2) maintains 4 models of the true 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[7] 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 maximising the score and minimising the number of states visited. Once that configuration was found we then used multiples of 2 to produce characteristic curves presented in the results.

[5] this may consume half of the resources in some games

[6] samples of the information set

[7] 4 x 2 x 10 x 4400, remember the Imperfect-Information Simulation starts at the beginning of the game, not the next round

Game Variants

The game variants were chosen to cast the strongest light on the experimental aims. We wanted to show the best possible performance for both players. With many game variations and player variations we chose configurations that gave a fair representation of the relative performances in context of the imperfect information present in each game.

Confidence Level

Each game variant and player configuration was played one thousand times[8] 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 nominated player scored the maximum payoff in every game.

A confidence level of 99% is shown using error bars in a two-tailed calculation of the standard error of the mean for both the states visited and the average payoff. That is to say that we were 99% confident that the player would score within the range reported using the computational resources reported. In some cases the error bars are smaller than the marker used to plot the point.

Equipment

The experiments were conducted on the high-throughput computer facilities at the School of Computer Science and Engineering at the University of New South Wales. That is, each experiment was distributed across several hundred laboratory PCs and the results collected and tabulated.

Experimental Results

We have conducted experiments on various configurations of players for each of the games reported. The lines joining points plotted in each chart do not represent a continuous function, but are used to link results for similar configurations of player and game.

Each point represents a different resourcing of that player in terms of the number of samples taken from the information set, and the number of playouts made when selecting a move. Typically resources were doubled from one configuration to the next, so a log scale is used for the horizontal axis of the chart.

Hidden Connect

In 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 purchasing moves the HP player performed as well as the HP-II player but consumed considerable fewer resources with both players improving their performance as they visited more states.

From the experimental results in Figure 3 we see the HP-II player required more than ten times the resources for a similar level of performance in the 3x3 grid, and 100 times the resources for the 5x5 grid. We also note that a the increase from 3x3 with a playout depth of 9 to the 5x5 with a depth

[8] for larger game variants fewer games were played and this is reflected in the error bars shown on the chart

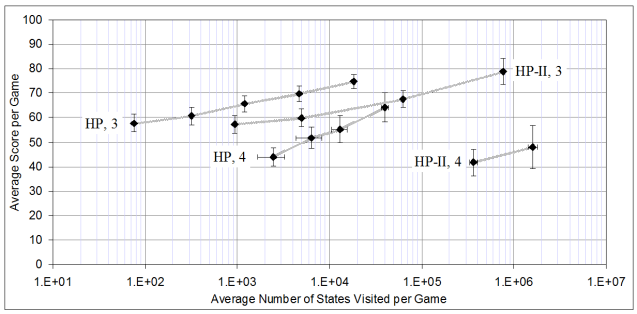

Figure 3: Hidden Connect game showing the HP player and the HP-II player performance

of 25 would give a predicted 13 fold[9] increase for the HP player and a 165 fold increase for the HP-II player. These predictions appear to be consistent with the experimental results.

Mastermind

In Mastermind we used two variants of the game: two colours in three positions with three guesses, and three colours in four positions with four guesses. The reward was pro rata for the number of correct positions. The number of guesses was restricted to see if an increase in resources would improve the guessing strategy. This game has no hidden game play, only a hidden initial setting created by the random player. As such, even the simplest HP player was able to solve the puzzle.

The HP-II player consumed considerable more resources for no additional improvement in performance.

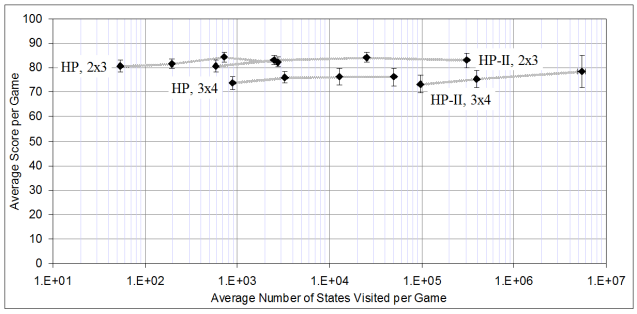

Figure 4: Mastermind game showing the HP player and the HP-II player performance

From the experimental results in Figure 4 we see a very flat performance from both players. This game is solved more in the backtracking of invalid samples, and less in the cleverness of the guesses. Whilst a skilled player might achieve a binary search, even the least resourced player was able to achieve an optimal result[10].

[9] from $O(bf \cdot d^2)$ for the HP player to $O(bf^2 \cdot d^4)$ for the HP-II player, if all games were played out to a draw

[10] given there were limited guesses

Number Guessing

In the Number Guessing Game 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 HP-II player tended towards optimum play as the resources were increased.

The performance of the HP player may seem unusual, but is not unexpected as it tends towards score for a random guesser. That is, when fully resourced it should score $1/n$. However, when poorly resourced it asks a few random questions and hence accidentally[11] narrows the field.

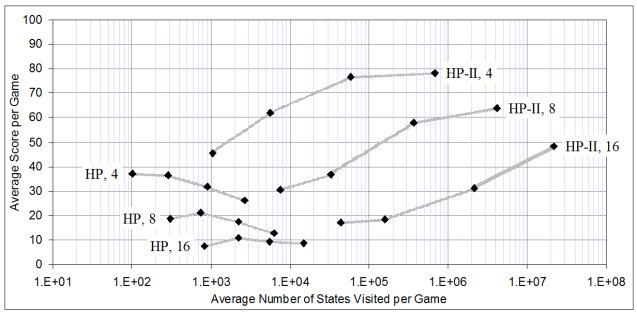

Figure 5: Number Guessing game showing the HP player and the HP-II player performance

From the experimental results in Figure 5 we see the HP-II player required significantly more resources than its HP counterpart. We also note that a doubling of the game size increased the computation resource requirements by several orders of magnitude for the same level of performance[12]. Without any increase in game play complexity we would expect a theoretical increase of 10 fold for a doubling of the game size.

Banker and Thief

In the Banker And Thief game we use variants with 2 and 4 banks and a deposits of 10 by $10.00. As expected the fully resourced HP banker uses a greedy strategy when making deposits and falls victim to the thief. Whereas, the HP-II player tended towards optimum play as the resources were increased[13].

From the experimental results in Figure 6 we see a relatively small shift from one game variant to the next as this game has a fixed depth and only variable branching factor. Theoretically we should see a factor of 2 for the HP player and 4 for the HP-II player from one game variant to the next. However, we are measuring backtracking states as well as playout states. We Also see that the HP-II player requires resources an order of magnitude more than its HP counterpart.

[11] from its point of view after elevating sample to fact

[12] optimal play with 4 number scores 80, 8 numbers scores 70 and 16 numbers scores 60

[13] optimum play rewards $40.00 by creating a false target of $60.00

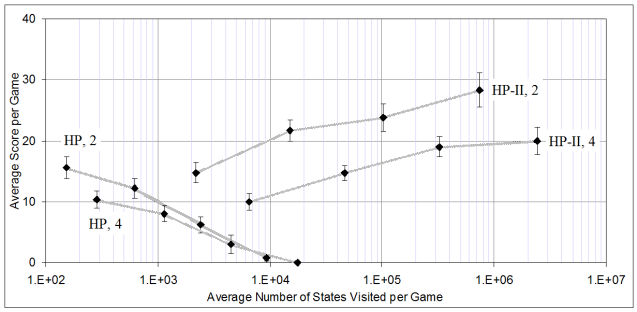

Figure 6: Banker and Thief game showing the HP player and the HP-II player performance

Battleships in Fog

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

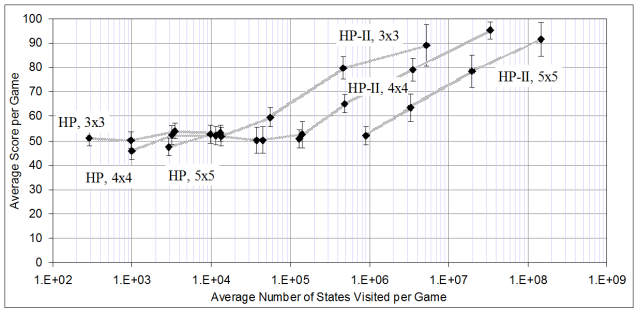

Figure 7: Battleships in Fog game showing the HP player and the HP-II player performance

From the experimental results in Figure 7 we see the HP-II player required significantly more resources than its HP counterpart. We also note that an effective doubling of the game size[14] increased the computation resource requirements by an order of magnitude for the same level of performance. Without any increase in game play complexity, and hence the length of the game, we would expect a theoretical increase of 4 fold for a doubling of the game size.

Discounting and Pruning

We implemented discounting and pruning for the HP-II player to guage its effectiveness. We chose a fully resourced version of the player so as to give the greatest opportunity for improvement.

For each of the games we implemented each of the variation of discounting and pruning and reported the average score and average number of states per game below.

[14] from 3x3=9 to 4x4=16

HP-II player

Game	Enhancement	Score	States
Hidden Connect (3x3)	None	78.8	768,687
	CoW	63.6	548,684
	Discounting	79.4	753,236
	PoD	52.7	520,416
Mastermind (2x3)	None	83.1	307,619
	CoW	81.2	68,300
	Discounting	84.1	303,373
	PoD	81.6	82,812
Number Guessing (4)	None	78.2	677,314
	CoW	70.6	677,190
	Discounting	78.3	672,235
	PoD	70.5	699,806
Banker & Thief (2x10)	None	28.3	743,705
	CoW	14.9	660,727
	Discounting	27.1	744,103
	PoD	26.7	744,071
Battleships in Fog (3x3)	None	89.1	5,236,047
	CoW	42.3	2,482,041
	Discounting	86.6	5,157,731
	PoD	48.8	3,027,023

Figure 8: Results of pruning the search space on player performance

Hidden Connect We used the 3x3 grid and saw only a slight reduction in states visited using pruning, but a significant degradation of performance.

Mastermind We used the two colour, three position version and saw a significant reduction in resource using pruning with no degradation of performance.

Number Guessing We used the 4 number version of this game and saw no real improvement from pruning, in fact pruning extended some games, reducing the score and increasing resources per game.

Banker and Thief We used the 2 bank, 10 deposit version of this game and saw only a small reduction in resources for Cut on Win, but a significant degradation of performance.

Battleships in the Fog We used the 3x3 grid for this game and saw a significant reduction in resources using pruning, but a significant degradation of performance.

Conclusions

In each of the game types we see confirmation of the two basic questions:

- Does HP-II perform better than HP at this type of game, and at what computational cost; and
- What is the impact of up-sizing the game on the computational cost for HP-II to achieve the same level of performance.

HP versus HP-II When the topology is favourable the HP player performs as well as the HP-II player, improving its score as resources increase and reaching the same level of optimal play. So we would conclude that the HP player is an acceptable choice, except where the game topology makes it ineffective.

Computation Cost of HP-II The HP-II player requires significantly more resources to instantiate than the HP player. In each of the games tested the number of states visited increased by an order of magnitude. The only benefit in using the HP-II player is that it correctly values information. So we conclude that the HP player should be the first choice, except where the game topology makes it ineffective.

Up-sizing the Game In all of the games tested we saw a significant impact when the game was up-sized. This was consistent with the theoretical analysis that stated the HP player as being $O(bf \cdot d^2)$ and the HP-II player as being $O(bf^2 \cdot d^4)$.

Discounting In all of the games discounting had little impact on the outcome. In games with fixed depth, discounting is known to have no impact. In the other games discounting did not hasten the win, or prolong the loss in any real way.

Pruning There was only one game out of five where pruning had a positive impact. Cut on Win and Pruning on Depth are known to be safe (Cazenave et al. 2016) for Nested Monte Carlo players with complete information. The results from Banker and Thief, and Battleships in Fog suggest they may not be safe in Imperfect-Information Simulations, but the reason is not clear[15].

General The HP-II player will always play as well as the HP player, and will correctly value information in the context of the reward structure and the expected outcome of the game. Whereas, the HP player falls into the trap of elevating sample to fact and consequently values information at zero.

The player of choice should be the HP player, only utilising the information valuing properties of the HP-II player when the game topology dictates.

Future Work

We see ample opportunity for extending this work. Specifically, looking for ways to reduce the resources consumed by the nested playouts in the HP-II player.

Another obvious area of interest is the safeness of the pruning technique used in Imperfect-Information Simulations.

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.

[15] samples of the information set may not contain the same legal moves, but to offer this as a reason would be speculation

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 *Proceedings of the International Conference on Computers and Games (CG)*, 21–34.

Browne, C.; Powley, E.; Whitehouse, D.; Lucas, S.; Cowling, P.; Rohlfshagen, P.; Tavener, S.; Perez, D.; Samothrakis, S.; and Colton, S. 2012. A survey of Monte Carlo Tree Search methods. *IEEE Transactions on Computational Intelligence and AI in Games* 4(1):1–43.

Cazenave, T.; Saffidine, A.; Schofield, M.; and Thielscher, M. 2016. Discounting and pruning for nested playouts in general game playing. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

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., and Björnsson, Y. 2013. The international general game playing competition. *AI Magazine* 34(2):107–111.

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. *Journal of Artificial Intelligence Research* 14:303–358.

Kupferschmid, S., and Helmert, M. 2007. A Skat player based on Monte-Carlo simulation. In *Proceedings of the International Conference on Computers and Games (CG)*, 135–147.

Long, J.; Sturtevant, N.; Buro, M.; and Furtak., T. 2010. Understanding the success of perfect information M Carlo sampling in game tree search. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 134–140. Atlanta: AAAI Press.

Richards, M., and Amir, E. 2009. Information set sampling for general imperfect information positional games. In *Proceedings of the IJCAI Workshop on General Intelligence in Game-Playing Agents (GIGA)*, 59–66.

Schiffel, S., and Thielscher, M. 2007. Fluxplayer: A successful general game player. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 1191–1196.

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., and Thielscher, M. 2015. Lifting model sampling for general game playing to incomplete-information models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 3585–3591.

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.

Wisser, F. 2015. An expert-level card playing agent based on a variant of perfect information Monte Carlo sampling. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 125–131. Buenos Aires: AAAI Press.