HyperPlay: A Solution to General Game Playing with Imperfect Information

M. Schofield, T. Cerexhe and M. Thielscher

Karpathiotaki Maria
M1279

karpathiotaki.maria@gmail.com

General Game Playing…

- Intelligent agents that can automatically learn how to skillfully play **a wide variety of games**, given only the descriptions of the game rules.
- Agents have to learn diverse game-playing strategies without any game-specific knowledge being provided by their developers.
- Relevant game-specific knowledge required for expert-level play, must be effectively discovered **during play**!!

…with Imperfect Information

- Players don't know exactly the actions chosen by other players.

- They know who the other players are, what their **possible** strategies/actions are, and the preferences/payoffs of these other players.

- Hence, information about the other players is imperfect.

    - E.g.: Card games, like bridge and poker.

GDL: Game Description Language

- A first-order logic based language (variant of Datalog) for defining discrete games with **complete information**!
- The expressiveness of GDL allows a large range of deterministic, perfect information, simultaneous-move games to be described, with any number of adversary or cooperating players.
  - **Turn-based** games are modeled by having the players who do not have a turn return a special no operation move.

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

GDL-II: GDL with incomplete/imperfect information

- GDL has recently been extended.
- Two new keywords to describe arbitrary (finite) games with **randomized** moves and **imperfect information**:
  - sees
  - random

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

HyperPlay: The Technique

- General approach that can be used by **any** general game player.
- The intuition is to **translate** imperfect-information games into a format suitable for simpler, perfect-information players.

HyperPlay: The Algorithm

- We maintain a bag *H* of **HyperGames** (random samples or "guesses" of the current true game state).
- In each round n, a perfect-information player can select a next move a_n suitable for each of these isolated models.
- Our move selection is then submitted to the game controller and a new set of percepts in for this round is received.
- Each model M in our bag of samples *H* is then **propagated** forward to reflect the deeper game tree.
- If we select a path that the last percepts reveal to be impossible, then we reach a state where no consistent joint move can be found, so:
  - we **backtrack** by adding the guilty move vector to a set of bad moves for that state,
  - and we call **forward** on this earlier game node, effectively undoing the move and attempting to push forward again.
- This process repeats until a consistent model is found for the current round.

Move Selection

- The HyperPlay algorithm is **agnostic** of the move selection process.

- Move selection should be based on:
  - the **expected** value of a move in a HyperGame
  - the **propability** that the HyperGame is the true game

Experiments

- Several games were selected:
  - Monty Hall,
  - Krieg-TicTacToe,
  - Blind BreakThrough
- The HyperPlayer opposed a **Cheat,** a HyperPlayer with access to the true game, and **fully resourced** so that it made the best move choices within the limitations of the move selection process.
  - The method for calculating the Cheat's resources was to play one Cheat against another Cheat with different resources.

Results

- The results showed a successful implementation of the HyperPlay technique for playing imperfect information games.
- The collection of models:
  - can be very **accurate**,
  - is a **credible substitute** for perfect information about the true game,
  - can be **competitive** even against a Cheat.

References

- Bjornsson, Y., and Finnsson, H. 2009. *CadiaPlayer: A simulation-based general game player*. IEEE Transactions on Computational Intelligence and AI in Games 1(1):4–15.
- 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. 2011. *GIB: Imperfect information in a computationally challenging game*. CoRR.
- 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.
- Rosenhouse, J. 2009. *The Monty Hall Problem*. Oxford University Press.
- **Schofield, M., Cerexhe, T. and Thielscher, M. 2012. *HyperPlay: A Solution to General Game Playing with Imperfect Information*. In Proc. AAAI, 1606-1612.**
- Thielscher, M. 2010. *A general game description language for incomplete information games*. In Proc. AAAI, 994–999.

Thank you!

Monty Hall

```
role(candidate).   role(random).

init(closed(1)).   init(closed(2)).   init(closed(3)).
init(step(1)).

legal(random,hide_car(?d))   <= true(step(1)),
                                true(closed(?d)).
legal(random,open_door(?d))  <= true(step(2)),
                                true(closed(?d)),
                                not true(car(?d)),
                                not true(chosen(?d)).
legal(random,noop)           <= true(step(3)).
legal(candidate,choose(?d))  <= true(step(1)),
                                true(closed(?d)).
legal(candidate,noop)        <= true(step(2)).
legal(candidate,noop)        <= true(step(3)).
legal(candidate,switch)      <= true(step(3)).

sees(candidate,?d)  <= does(random,open_door(?d)).
sees(candidate,?d)  <= true(step(3)),  true(car(?d)).
next(car(?d))      <= does(random,hide_car(?d)).
next(car(?d))      <= true(car(?d)).
next(closed(?d))   <= true(closed(?d)),
                      not does(random,open_door(?d)).
next(chosen(?d))   <= does(candidate,choose(?d)).
next(chosen(?d))   <= true(chosen(?d)),
                      not does(candidate,switch).
next(chosen(?d))   <= does(candidate,switch),
                      true(closed(?d)),
                      not true(chosen(?d)).

next(step(2))  <= true(step(1)).
next(step(3))  <= true(step(2)).
next(step(4))  <= true(step(3)).

terminal  <= true(step(4)).

goal(candidate,100)  <= true(chosen(?d)),  true(car(?d)).
goal(candidate,  0)  <= true(chosen(?d)),  not true(car(?d)).
goal(random,0).
```

A GDL – II description of the Monty Hall game [Rosenhouse, 2009] adapted from [Thielscher, 2011].

HyperPlay: The Algorithm

```
procedure main()
begin
  H := {|⟨∅⟩, . . . , ⟨∅⟩|}
  n := 1
  repeat
    a_n := select_move(H)
    I_n := submit_move(a_n)
    for all M ∈ H do
      forward(M, n + 1)
    n := n + 1
  until end_of_game
end

procedure forward(M = ⟨B_1, m⃗_1, . . . , B_{k−1}, m⃗_{k−1}, B_k⟩, n)
begin
  if k < n then
    if choose m⃗ ∈ L(M) \ B_k
           with m⃗|_r = a_k && I(m⃗, M) = I_k then
      M := M ⊕ m⃗
      forward(M, n)
    else
      backtrack(M, n)
end

procedure backtrack(⟨B_1, m⃗_1, . . . , B_{k−1}, m⃗_{k−1}, B_k⟩, n)
begin
  B_{k−1} := B_{k−1} ∪ {m⃗_{k−1}}
  forward(⟨B_1, m⃗_1, . . . , B_{k−1}⟩, n)
end
```

HyperPlay [Schofield, 2012].

Move Selection

- $P(HG_i | Percepts) = \frac{P(Percepts | HG_i) * P(HG_i)}{P(Percepts)}$

- $P(HG_i | Percepts) \sim P(HG_i)$

Move Selection

- $ChoiceFactor_i = \prod_j Choices_{i,j}$

- $$P(HG_i) = \frac{1/\ ChoiceFactor_i}{\sum_n 1/\ ChoiceFactor_n}$$

- $E\ (Move_j) = \sum E(Move_{i,j}) * P(HG_i)$