# Inductive Programming
# Lecture 8
# Game Strategy Induction

Stephen Muggleton
University of Nanjing and
Emeritus Professor Imperial College London

30th November, 2025

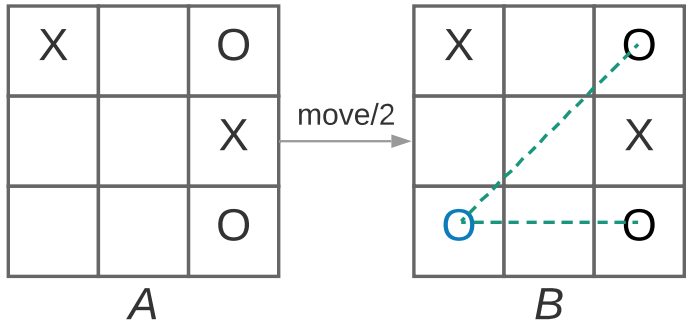
## Papers for this lecture

**Paper8.1:** S.H. Muggleton and C. Hocquette. Machine discovery of comprehensible strategies for simple games using meta-interpretive learning. New Generation Computing, 37:203-217, 2019.

## Motivation

- Inductive Programming and AI
- World-class play for Go, Chess, Checkers - AlphaGo (2016) and AlphaZero (2018)
- Deep Reinforcement Learning - played 30 (AlphaGo) million to 44 million (AlphaZero) self-play games
- Go has more than $10^{108}$ possible game sequences. Observable Universe around $10^{80}$ elementary particles, and $10^{16}$ seconds old
- Poor Data Efficiency and Human Comprehensibility
- Meta-Interpretive Game Ordinator (MIGO)
- Minimax Evaluable games - Noughts-and-Crosses and Hexapawn

## Noughts and Crosses




```
win_2(A,B):-win_2_1_1(A,B),not(win_2_1_1(B,C)).

win_2_1_1(A,B):-move(A,B),not(win_1(B,C)).

win_1(A,B):- move(A,B),won(B).
```

## Related work

**Reinforcement Learning** World's first reinforecment learning, MENACE (Michie, 1963) learned noughts-and-crosses using matchboxes, punishment and reward beads. HER (Gardner, 1962) for Hexapawn.

**Chess endgame strategies** Learn minimax depth-of-win using ID3 (Shapiro,Niblett, 1982; Quinlan, 1983) and ILP (Bain Muggleton, 1995).

**Q-learning** Learn optimal policy (Watkins, 1989). Asymptotic convergence proved (Watkins,Dayan, 1992).

**Relational Reinforcement Learning** States and actions represented relationally (Dzeroski et al, 2001). Single agent learning problems.

**Deep Q-learning** Extension of Q-learning with deep convolutional neural network (Mnih et al, 2015). Atari 2600 games. Also AlphaGo (Silver et al, 2016) and AlphaZero (Silver et al, 2018).

## Credit assignment problem

**Learning by playing** Learner evaluates success from outcomes of games.

**Credit assignment** What is reward for individual moves?

**Reinforcement Learning** Assign reward to individual moves based on a delay function. Rewards used to update parameters across all board states in game. The number of board states for Noughts-and-Crosses is $10^5$; Chess is $10^{45}$; Go is $10^{100}$.

**Exploration vs exploitation** Step size $\in [0, 1]$ is degree new information overides old.

**Discount factors** $\gamma \in [0, 1]$ is importance of future rewards.

**Function approximation** Deal with larger problem by approximating function over a continuous state space. eg using Convolution Neural Network.

## Credit assignment - MIGO

**Outcome** $Outcome(P, G) \in \{won, drawn, lost\}$ where $won \succ drawn \succ lost$

**Play** Learner $P_1$ plays against opponent $P_2$ which follows minimax strategy.

**Selection** Game starts from a randomly chosen initial board $B$.

**Lemma 1** The outcome of $P_1$ monotonically decreases during a game.

**Theorem 2** If the outcome is won for $P_1$, then every move of $P_1$ is a positive example for the task of winning.

**Theorem 3** If $S_W$ accurate strategy and $Outcome(S_W, G) \neq won$ and $Outcome(P_1, G) = drawn$ then every move of $P_1$ is a positive example for the task of drawing.

## MIGO algorithm - Dependency Learning

**Input**: Positive examples for win_k and draw_k

**Output**: Strategy for win_k and draw_k

```
1: for k in [1,Depth] do
2:     for each example of win_k/2 do
3:         one shot learn a rule and add it to the BK
4:     end for
5:     Learn win_k/2 and add it to the BK
6: end for
7: for k in [1,Depth] do
8:     for each example of draw_k/2 do
9:         one shot learn a rule and add it to the BK
10:    end for
11:    Learn draw_k/2 and add it to the BK
12: end for
```

## MIL representation

**Metarules**

| Name | Metarule |
|---|---|
| *postcond* | $P(A,B) \leftarrow Q(A,B), R(B).$ |
| *negation* | $P(A,B) \leftarrow Q(A,B), not(R(B,C)).$ |

**Board state**

| |
|---|
| Pair $s(B,P)$ where board $B$ and player $P$. |

**Primitives**

| Predicate | Call |
|---|---|
| Move | move$(S_1, S_2)$ |
| Won | won$(S)$ |
| Drawn | drawn$(S)$ |

## Game evaluation - minimax regret

**Defn 3.4** The **minimax regret** of game $G$ is the difference between minimax outcome of the initial position in $G$ and actual outcome of $G$.

**Cumulative minimax regret** The sum of minimax regret over a sequence of games. This is an objective measure of performance for competing strategies.

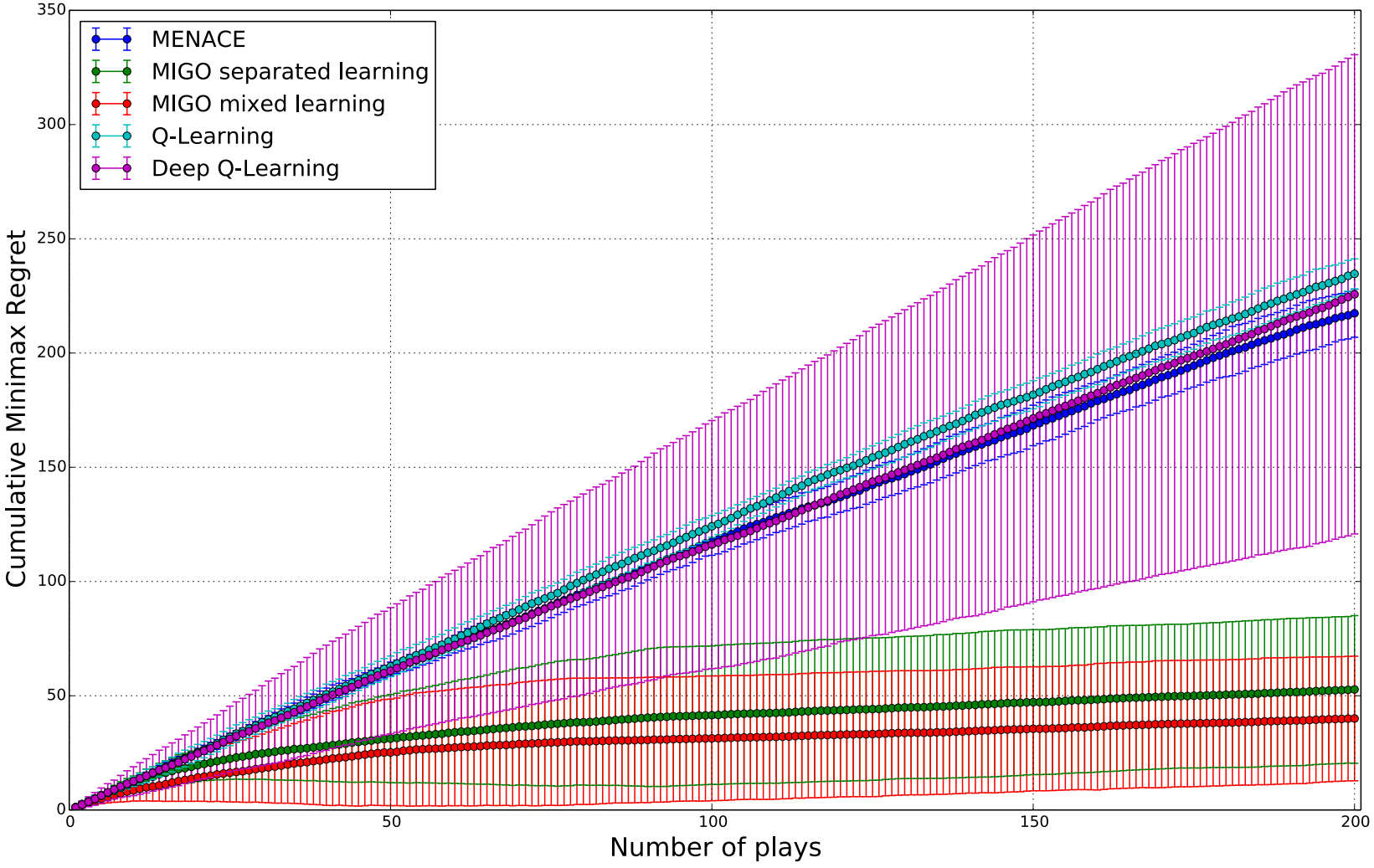
**Database** Minimax database computed beforehand.

**Experiment 1 - Comparison**
**Cumulative Minimax Regret**

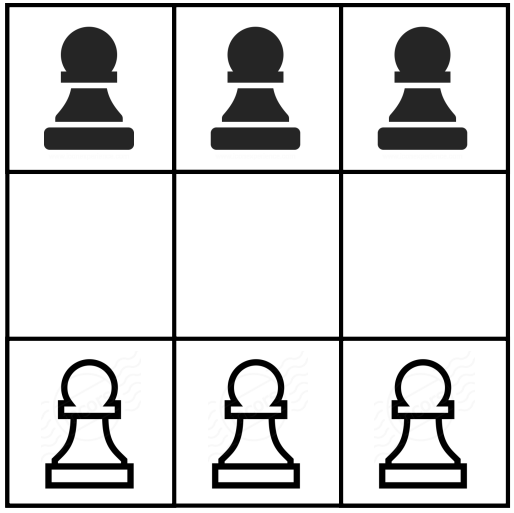
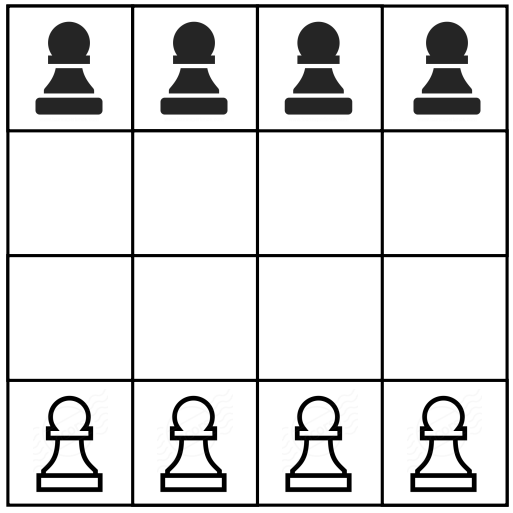
**Null Hypothesis 1** MIGO cannot converge faster than MENACE/HER, Q-learning and Deep Q-learning for learning optimal two-player game strategies.

Code for these experiments available at
https://github.com/migo19/migo.git

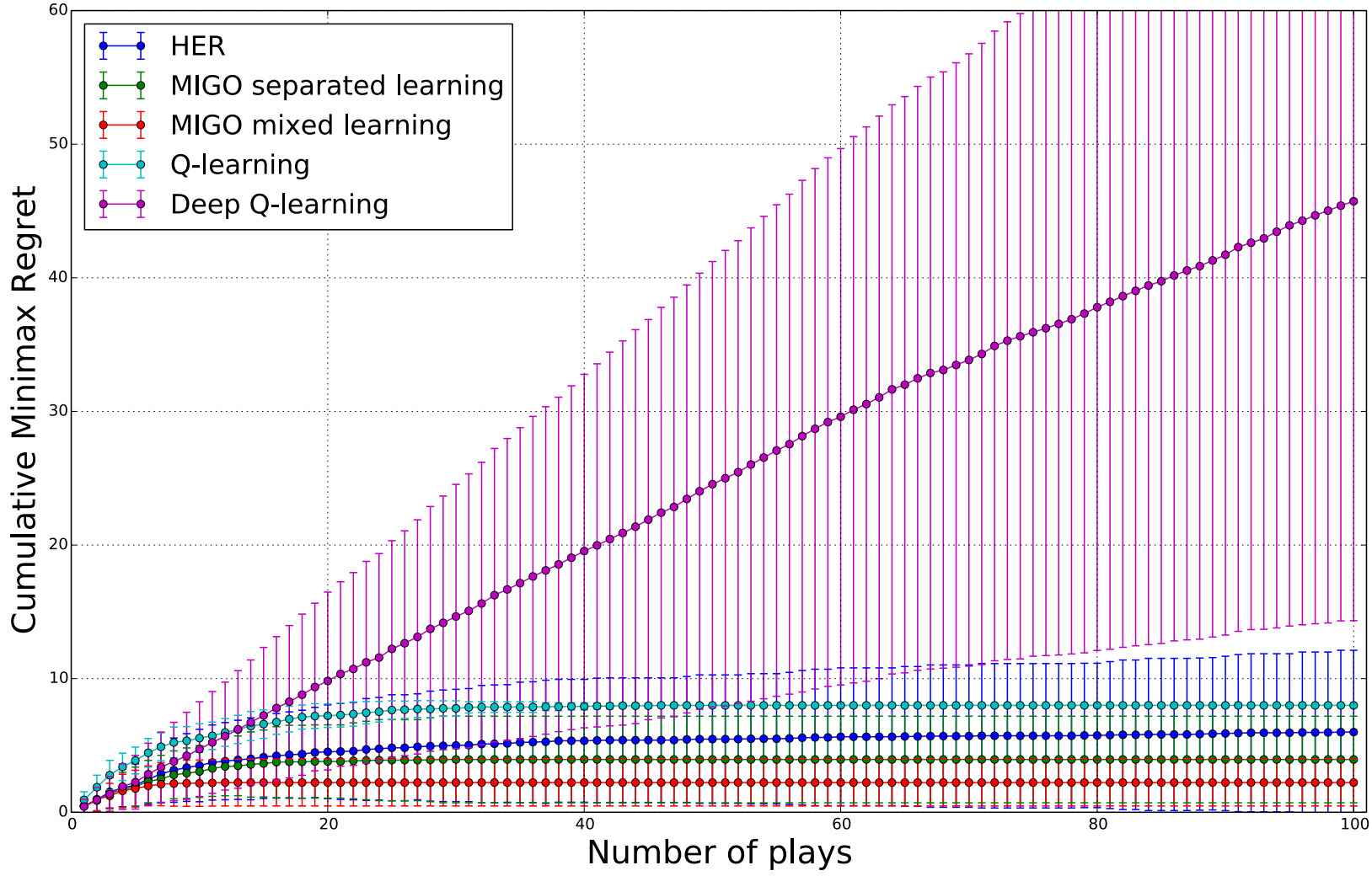
# Experiment 1 Nought-and-Crosses



# Hexapawn

| Hexapawn$_3$ | Hexapawn$_4$ |
| --- | --- |
|  |  |

# Experiment Hexapawn$_3$



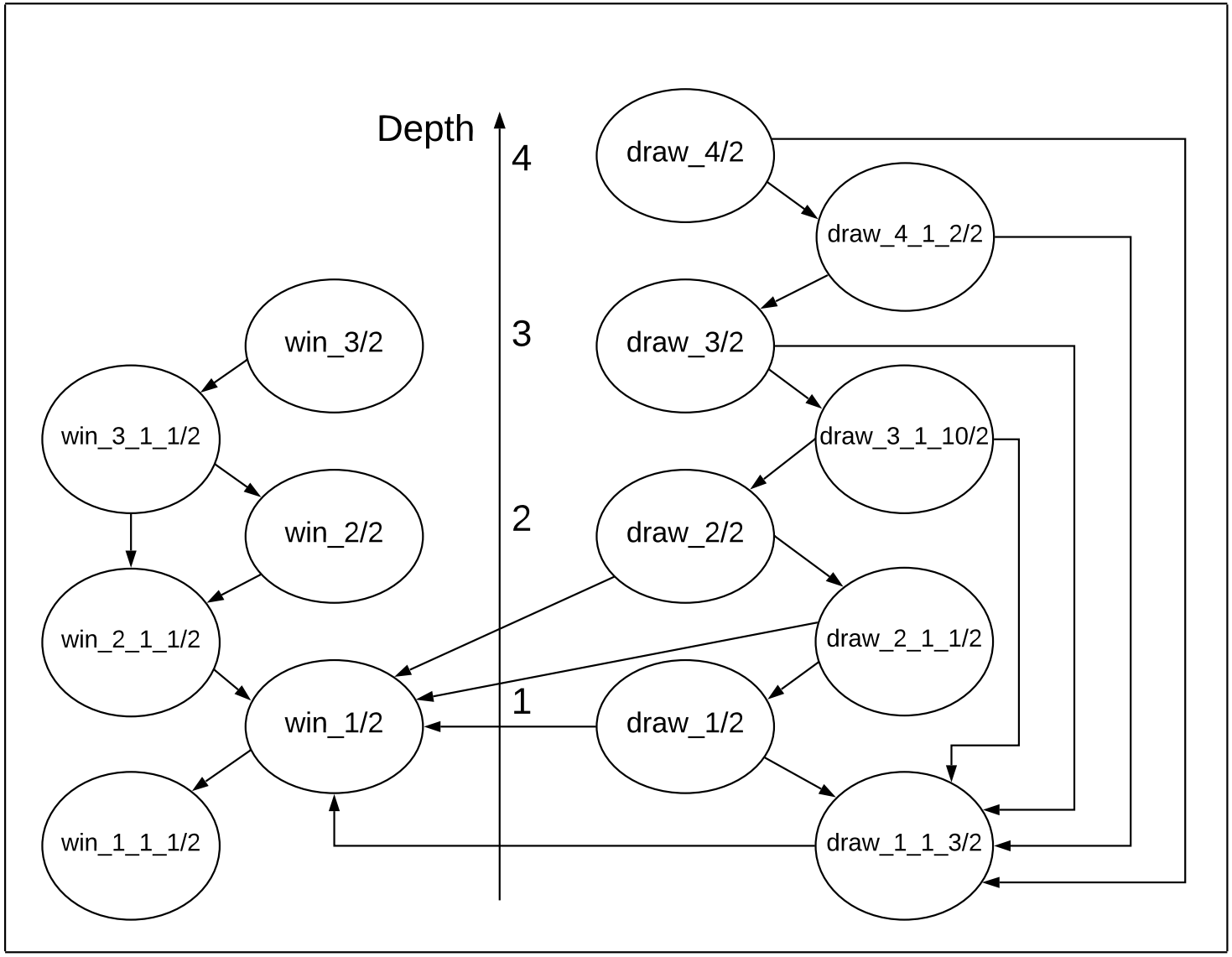
**Mean CPU seconds per iteration**

| | **OX** | **Hexapawn$_3$** | **Hexapawn$_4$** |
|---|---|---|---|
| MIGO mixed learning | $1.5.10^{-1}$ | $3.0.10^{-3}$ | 3.9 |
| MIGO separated learning | $8.9.10^{-2}$ | $2.8.10^{-3}$ | 3.8 |
| MENACE / HER | $1.5.10^{-3}$ | $2.7.10^{-4}$ | / |
| Q-Learning | $2.3.10^{-1}$ | $1.9\ .10^{-3}$ | $2.7\ .10^{-1}$ |
| Deep Q-Learning | $2.4.10^{-1}$ | $1.7.10^{-2}$ | $2.1\ .10^{-1}$ |

## Learned rules

| Depth | Rule |
|---|---|
| 1 | win_1(A,B):-win_1_1_1(A,B),won(B).<br>win_1_1_1(A,B):-move(A,B),won(B). |
| | draw_1(A,B):-draw_1_1_3(A,B),not(win_1(B,C)).<br>draw_1_1_3(A,B):-move(A,B),not(win_1(B,C)). |
| 2 | win_2(A,B):-win_2_1_1(A,B),not(win_2_1_1(B,C)).<br>win_2_1_1(A,B):-move(A,B),not(win_1(B,C)). |
| | draw_2(A,B):-draw_2_1_1(A,B),not(win_1(B,C)).<br>draw_2_1_1(A,B):-draw_1(A,B),not(win_1(B,C)). |
| 3 | win_3(A,B):-win_3_1_1(A,B),not(win_3_1_1(B,C)).<br>win_3_1_1(A,B):-win_2_1_1(A,B),not(win_2(B,C)). |
| | draw_3(A,B):-draw_3_1_10(A,B),not(draw_1_1_12(B,C)).<br>draw_3_1_10(A,B):-draw_2(A,B),not(draw_1_1_12(B,C)). |
| 4 | draw_4(A,B):-draw_4_1_2(A,B),not(draw_1_1_12(B,C)).<br>draw_4_1_2(A,B):-draw_3(A,B),not(draw_1_1_12(B,C)). |

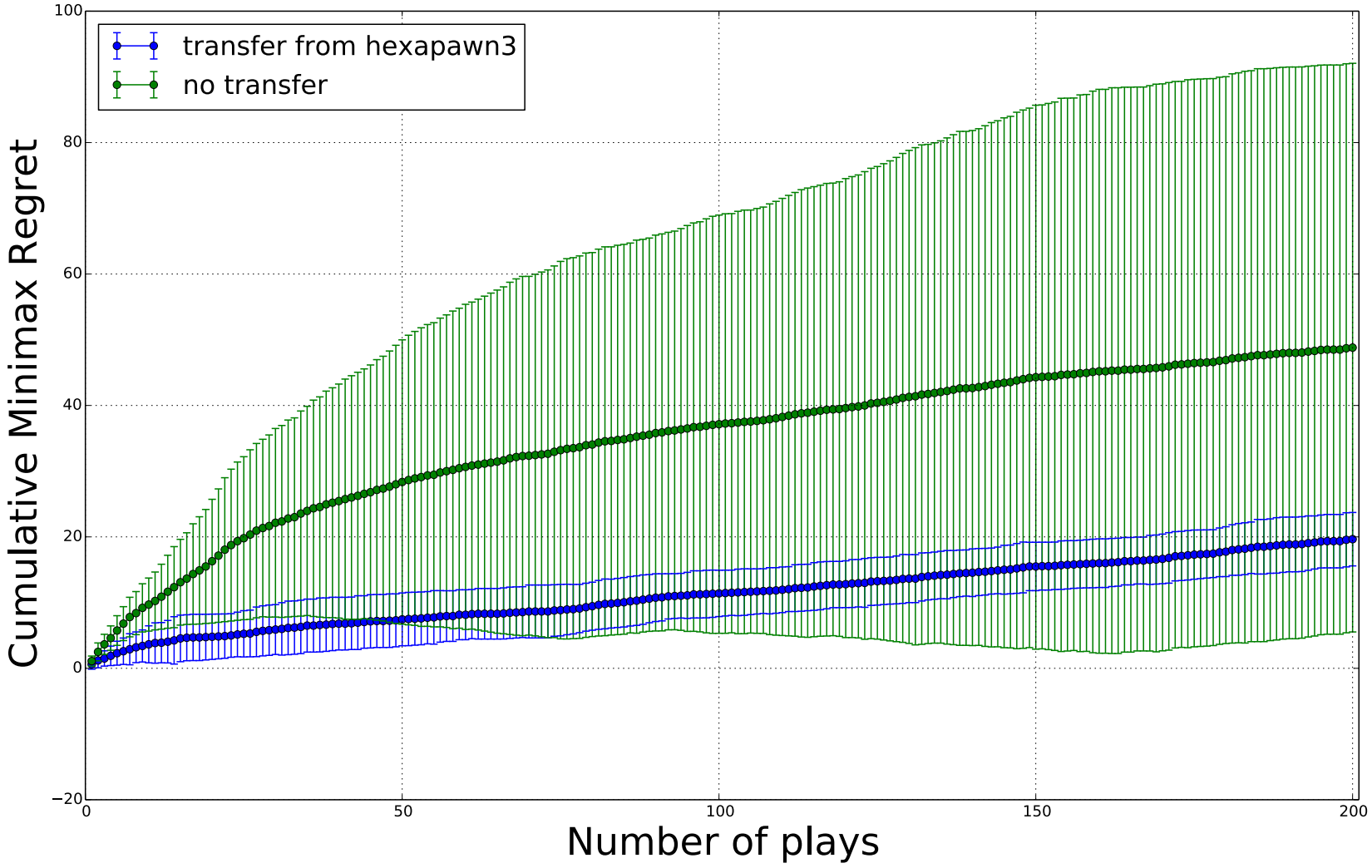
# Calling diagram



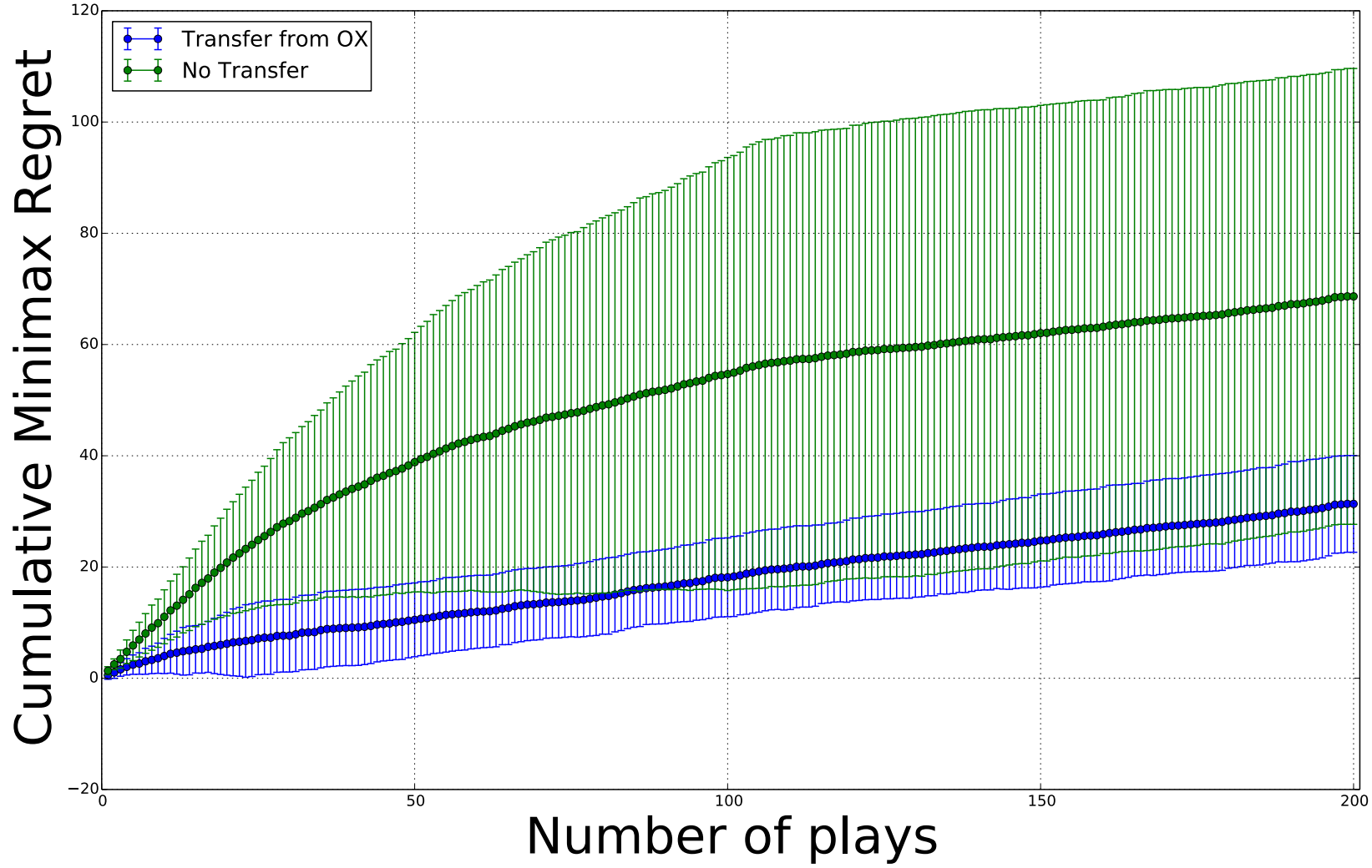
## Experiment 2

**Null Hypothesis 2** MIGO cannot transfer the knowledge learned during a previous task to a more complex game.

# Experiment 2a - Transfer Learning
# Hexapawn$_3$ to Noughts and Crosses



# Experiment 2b - Transfer Learning
# Noughts and Crosses to Hexapawn$_4$



**Summary**

- MIGO Meta-Interpretive Inductive Programming for two-player-games.
- Novel approach to Credit Assignment Problem.
- Lower Cumulative Minimax Regret than to Deep and classic Q-Learning.
- Strategies transferable to more complex games.
- Over-generalisation since learning from positive example only.
- Running time scales badly with large numbers of board states.
- Optimise running times using Metaopt.
- Assumes optimal opponent - relax assumptions and use self-play.
- Need to assess comprehensibility of strategies. Michie's Ultra-Strong Machine Learning.