Perfect information stochastic game playing : study of Carcassonne

Aymeric Behaegel, Quentin Cohen Solal[1], Tristan Cazenave [1]

1 : LAMSADE, Paris Dauphine

1. Monte Carlo Tree Search and UCT
2. AlphaZero and randomness
3. Carcassonne
4. Current work and possible improvements
5. References

Upper Confidence bound applied to Trees (UCT) and Monte Carlo Tree Search (MCTS)

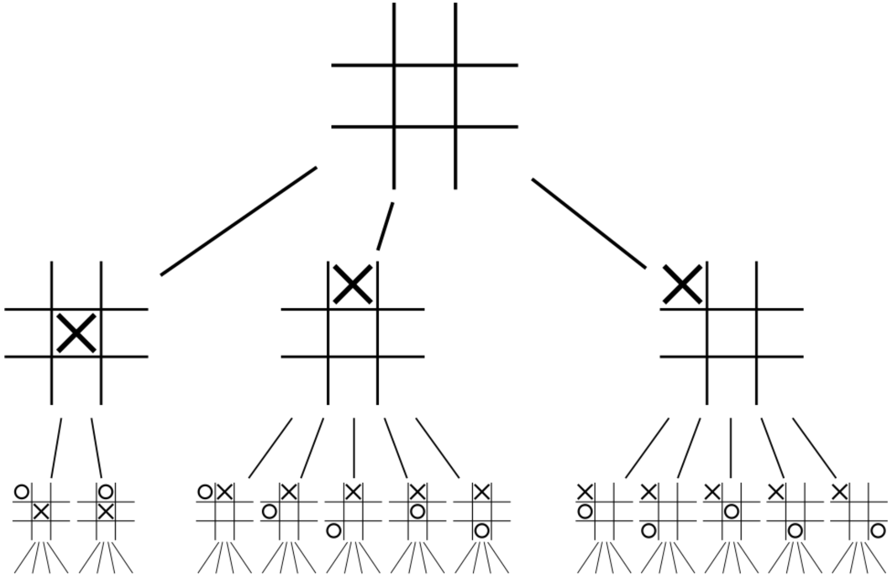
Monte Carlo Tree Search (MCTS)

Represent a **game** as a **tree**.

Each **node** represents a **state** of the game (with its value) and the directed **edges** are **moves** done by players.

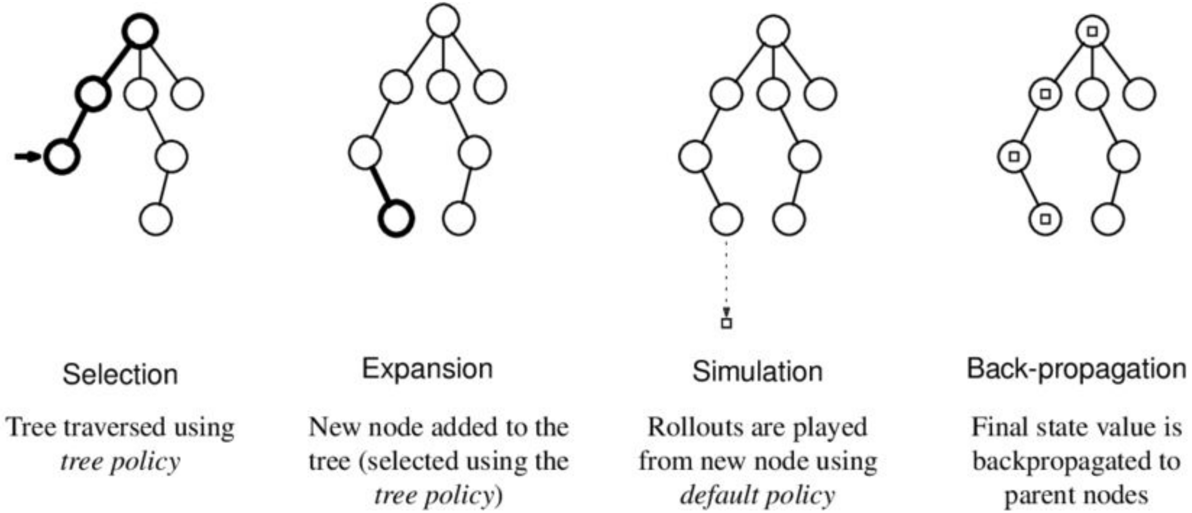
Explore the tree to find the **optimal play.**

Monte Carlo Tree Search (MCTS)



https://en.wikipedia.org/wiki/Game_tree#/media/File:Tic-tac-toe-game-tree.svg

Monte Carlo Tree Search (MCTS)



A. Santos, P. A. Santos and F. S. Melo, "Monte Carlo tree search experiments in hearthstone," *2017 IEEE Conference on Computational Intelligence and Games (CIG)*

Upper confidence bound applied to Trees (UCT)

$$UCT(node_i) = \frac{w_i}{n_i} + c\sqrt{\frac{\ln N_i}{n_i}}$$

- w_i = number of victories
- n_i = number of time the node has been visited
- N_i = number of time the parent node has been visited

jyopari, MCTS

Deep MCTS : AlphaZero

Deep MCTS : AlphaZero

Replace simulation by a single neural network with two heads :

- a value head : $v(s)$
- a policy head : $P(s, a)$

New formula : $$U(s, a) = Q(s, a) + c_{puct} * P(s, a) * \frac{\sqrt{\sum_b N(s,b)}}{1+N(s,a)}$$

with $Q(s, a) = \frac{N(s,a)*Q(s,a)+v(s)}{N(s,a)+1}$

AlphaZero : learning through self play

We then train the network by memorizing the training examples $(s_t, \vec{\pi}_t, z)$

with $\vec{\pi}_t$ being the MCTS policy vector, z the end result of the game, and the loss :

$$l = \sum_t (v_\theta(s_t) - z_t)^2 - \vec{\pi}_t * log(\vec{p}_\theta(s_t))$$

Stochastic AlphaZero

Chance nodes are added in between min and max player to represent the environment/randomness.

Can **greatly increase the branching factor** depending on the number of random possibilities.

Carcassonne

Carcassonne

Turn by turn, board constructing game

3 phases :

- draw and place a random tile on the board
- place meeple (or not) on the last tile
- count points / retract meeples

Goal : build (complete) roads, cities, churches, …

while possessing them to earn points

Carcassonne

Important informations :

- possible **tile positions** : 35*35*4 = 4900 (theoretical)
- in practice only a dozen possible action maximum
- possible **meeple placement** : 9 (if last tile known)
- 73 tiles (in the original game) to draw from the deck

↪ high branching factor, with **5*10^40 possible states**

Carcassonne : network input

Represent a state of the game as **22 channels** of 35*35*9:

- 5 channels for the board (cities, roads, monasteries, fields and shields)
- 5 channels for the next tile (same)
- 2*4 channels for placed meeples (number of players times type of terrain)
- 2 channels for free meeples
- 2 channels for phase

Current work and future improvements

Parallelization

Many **different types** of parallelization for MCTS. Some not applicable to AlphaZero.

Can accelerate **training** by playing games in parallel and producing the data at the same time.

Same model between epochs, so **no data sharing issues**.

Further improvements : MuZero

Model-based algorithm : transform an observation into a **hidden state** to only retain important informations. Navigate the “tree” with hidden states and hypothetical actions.

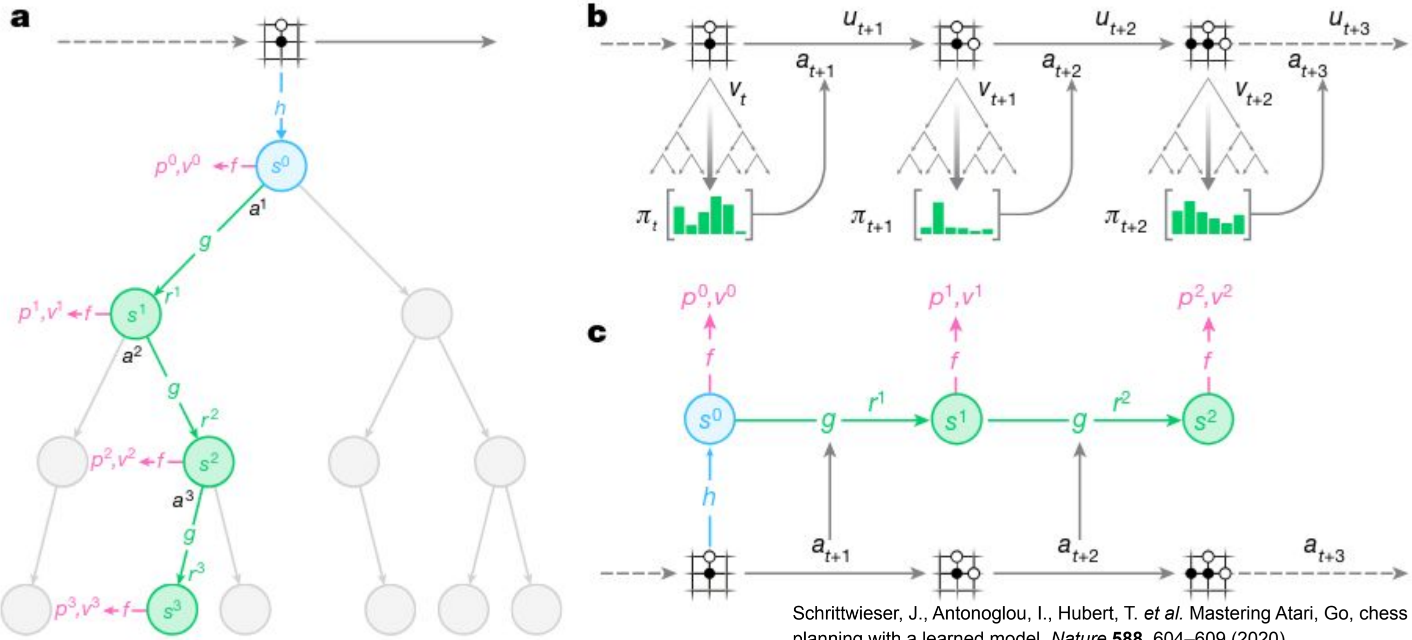
Useful for **complex games**, with complex mechanics that are hard/long to compute.

Further improvements : MuZero

At every step the model predicts the **policy**, the **value function** and the **immediate reward** of the hidden states through 3 functions :

- the *representation* function : $h_\theta(o_1, \ldots, o_t) = s^0$
- the *prediction* function : $f_\theta(s^k) = P^k, v^k$
- the *dynamics* function : $g_\theta(s^{k-1}, a^k) = r^k, s^k$

Further improvements : MuZero



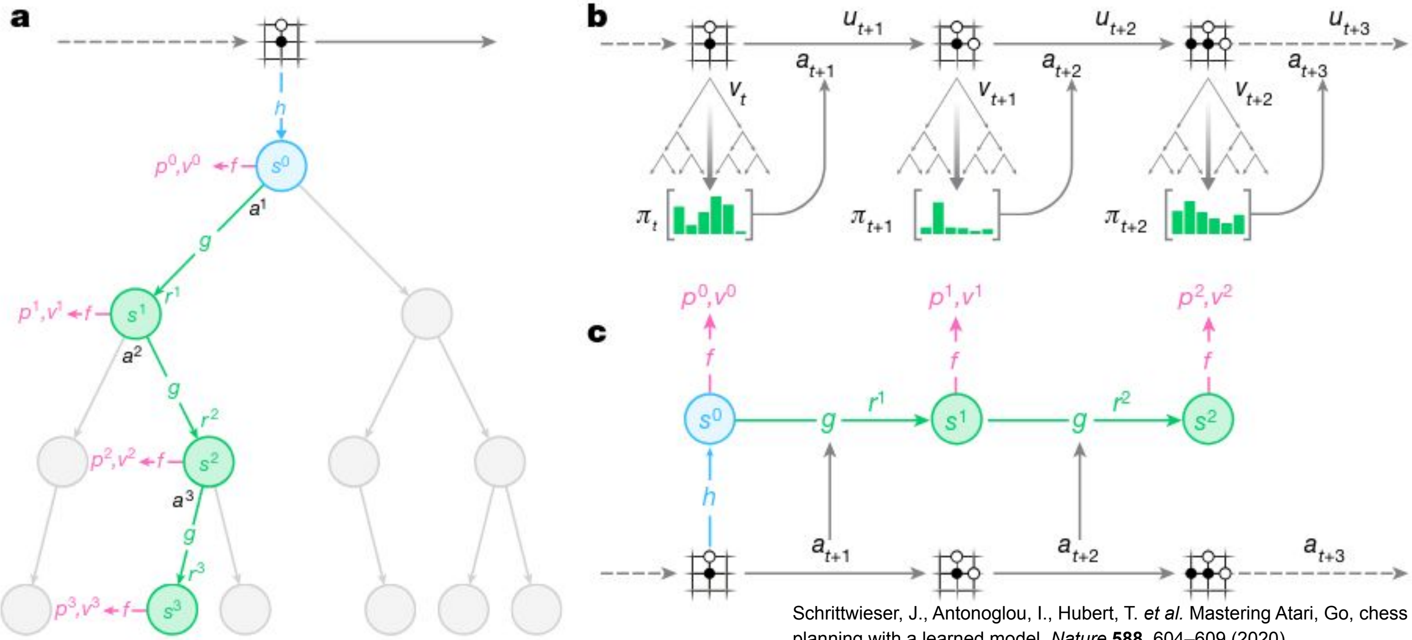
Schrittwieser, J., Antonoglou, I., Hubert, T. *et al.* Mastering Atari, Go, chess and shogi by planning with a learned model. *Nature* **588**, 604–609 (2020)

Further improvements : MuZero

The **policy**, the **value function** and the **immediate reward** are the three quantities that are trained to be predicted correctly through a replay buffer, they try to approximate the following quantities :

- $P^k \approx \pi_{t+k}$
- $v^k \approx z_{t+k}$, where $z_t = u_{t+1} + \gamma u_{t+2} + \cdots + \gamma^{n-1} u_{t+n} + \gamma^n v_{t+n}$
- $r_{t+k} \approx u_{t+k}$, where u_{t+k} is the immediate reward

Further improvements : MuZero



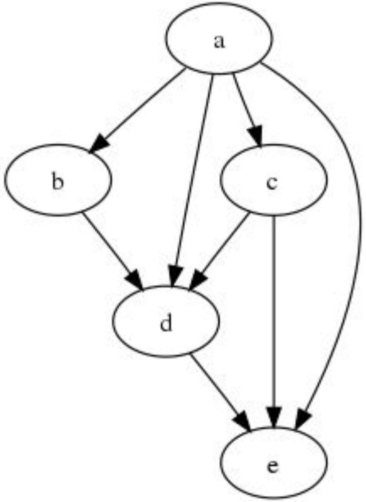
Schrittwieser, J., Antonoglou, I., Hubert, T. *et al.* Mastering Atari, Go, chess and shogi by planning with a learned model. *Nature* **588**, 604–609 (2020)

Further improvements : stochastic MuZero

Introduces **after-states**, to act as chance nodes (after an action is done, in between two states).

Only need to learn **after-states** and **chance outcomes** in order to generalize to stochastic games.

Upper Confidence bound for Directed acyclic graph (UCD)

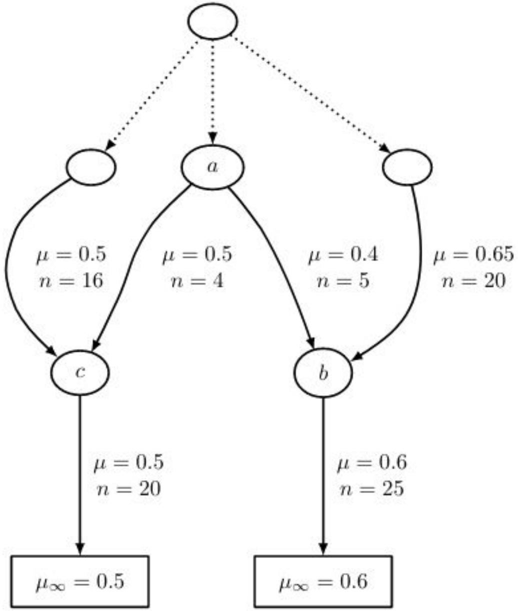


Wikipedia, Directed acyclic graph

Upper Confidence bound for Directed acyclic graph (UCD)

Need to adapt the UCT algorithm to DAGs.

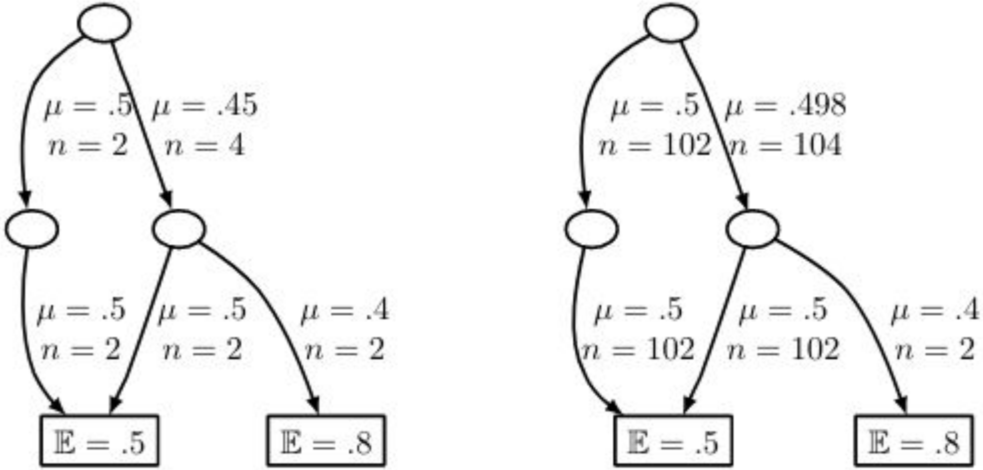
For example the backpropagation is no longer trivial : if left as is, we may find ourselves with a lack of information.



Cazenave, Tristan; Méhat, Jean; Saffidine, Abdallah (2012), *UCD : Upper confidence bound for rooted directed acyclic graphs*, Knowledge-Based Systems

Upper Confidence bound for Directed acyclic graph (UCD)

On the other hand if we update all stats from every possible path leading to a leaf, we end up with false conclusions.



(a) Initial settings (b) 100 playouts later

Cazenave, Tristan; Méhat, Jean; Saffidine, Abdallah (2012), *UCD : Upper confidence bound for rooted directed acyclic graphs*, Knowledge-Based Systems

Upper Confidence bound for Directed acyclic graph (UCD)

The solution found is an in-between : we backpropagate through the whole path plus all the possible path for a distance *d* above the leaf.

Upper Confidence bound for Directed acyclic graph (UCD)

Problem : UCD made for transpositions and not for imperfect information games. The DAG is not adapted for backpropagating impossible nodes/path.

Solution : we keep a tree as well as a DAG linked together by a transposition function; we navigate in the tree during the selection process (to avoid impossible states) and we use the DAG when we need informations.

References

References

- M. Świechowski and T. Tajmajer, "A Practical Solution to Handling Randomness and Imperfect Information in Monte Carlo Tree Search," *2021 16th Conference on Computer Science and Intelligence Systems (FedCSIS)*,2021
- Silver, David et al. "Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm." *ArXiv* abs/1712.01815 (2017): n. pag.
- Czech, Johannes et al. "Monte-Carlo Graph Search for AlphaZero." *ArXiv* abs/2012.11045 (2020): n. pag.
- Schrittwieser, J., Antonoglou, I., Hubert, T. *et al.* Mastering Atari, Go, chess and shogi by planning with a learned model. *Nature* **588**, 604–609 (2020). https://doi.org/10.1038/s41586-020-03051-4
- Antonoglou, Schrittwieser et al. "Planning in Stochastic Environments with a Learned Model", International Conference on Learning Representations, 2022