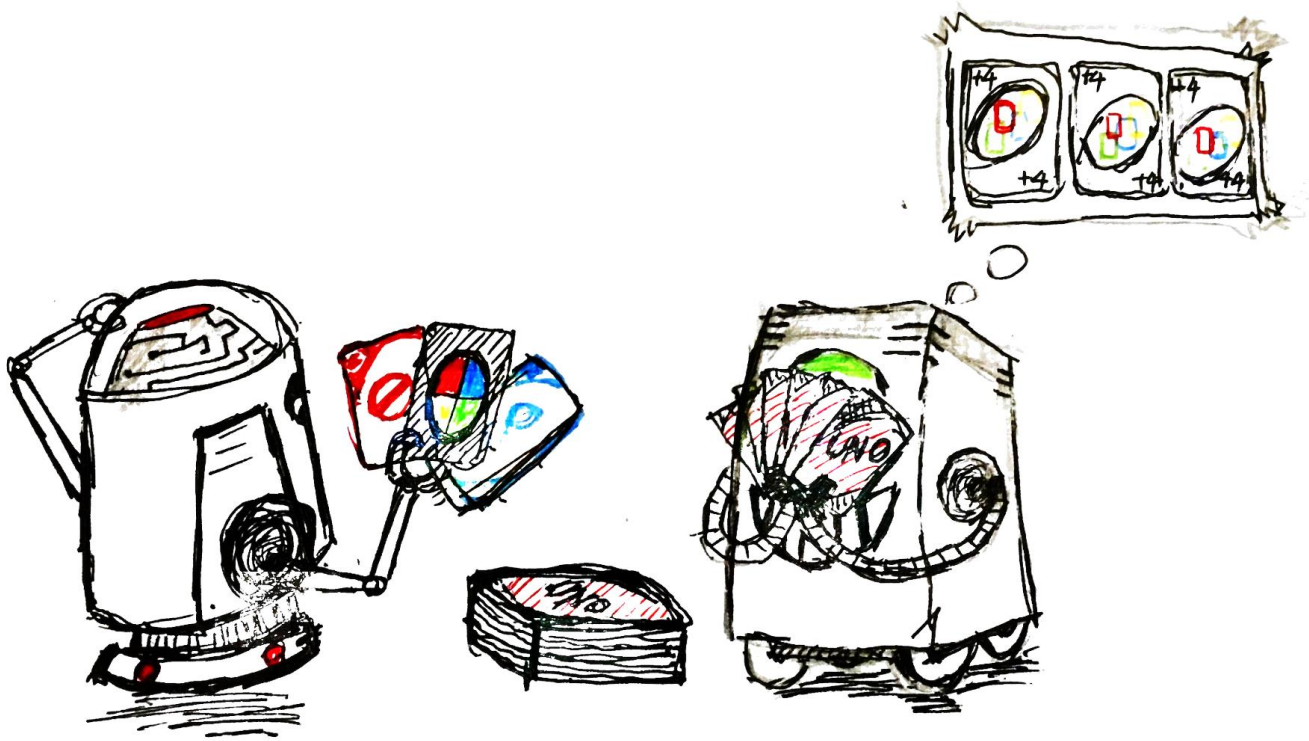
CS 181 Final Project

Monte Carlo in Monte Carlo: Pathway to Victory in UNO

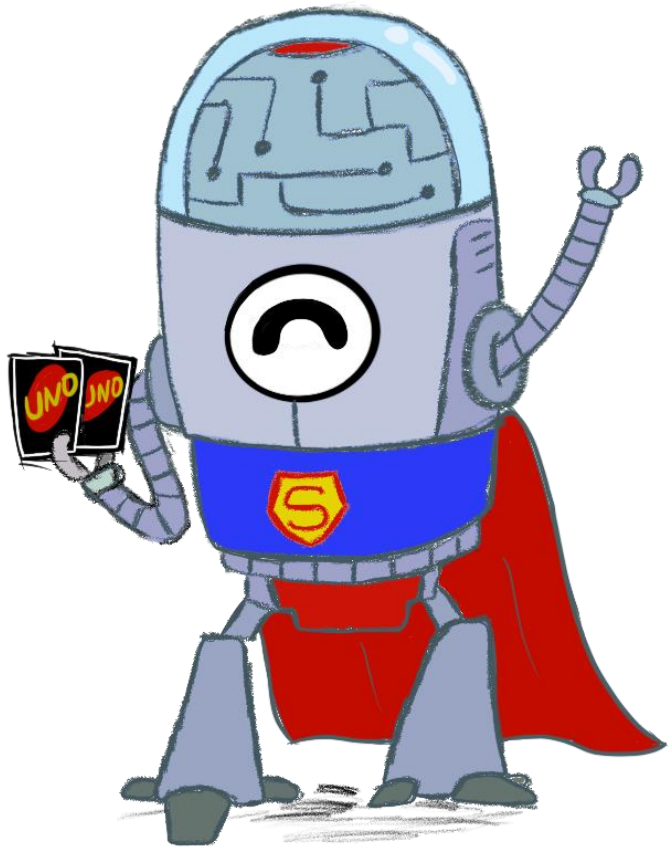


Wenyang Hui, Mengyun Liu, Chibin Zhang

[Slide template from Dan Klein and Pieter Abbeel for CS188 Intro to AI at UC Berkeley (ai.berkeley.edu).]

Problem Restatement

- How to achieve higher win rate against random agent strictly under the game rules of UNO.



UNO Rules

- Play the card of the same color or number as the previous played one.
- Wild cards can be played after any cards and assign color of the following card.
- If an agent is not able to play any cards, one card should be drawn.
- An agent wins if it has played all the cards.

Prev:

Contributions

- Several agents with excellent performance.
  - **MCTSAgent**, **ExpectimaxAgent**, DQNAgent and ReflexAgent
- Open loop method in imperfect information game.
  - Game uncertainty, hidden states
- Monte Carlo based method for predicting opponents' hands.
- A simple GUI

Baseline Agents

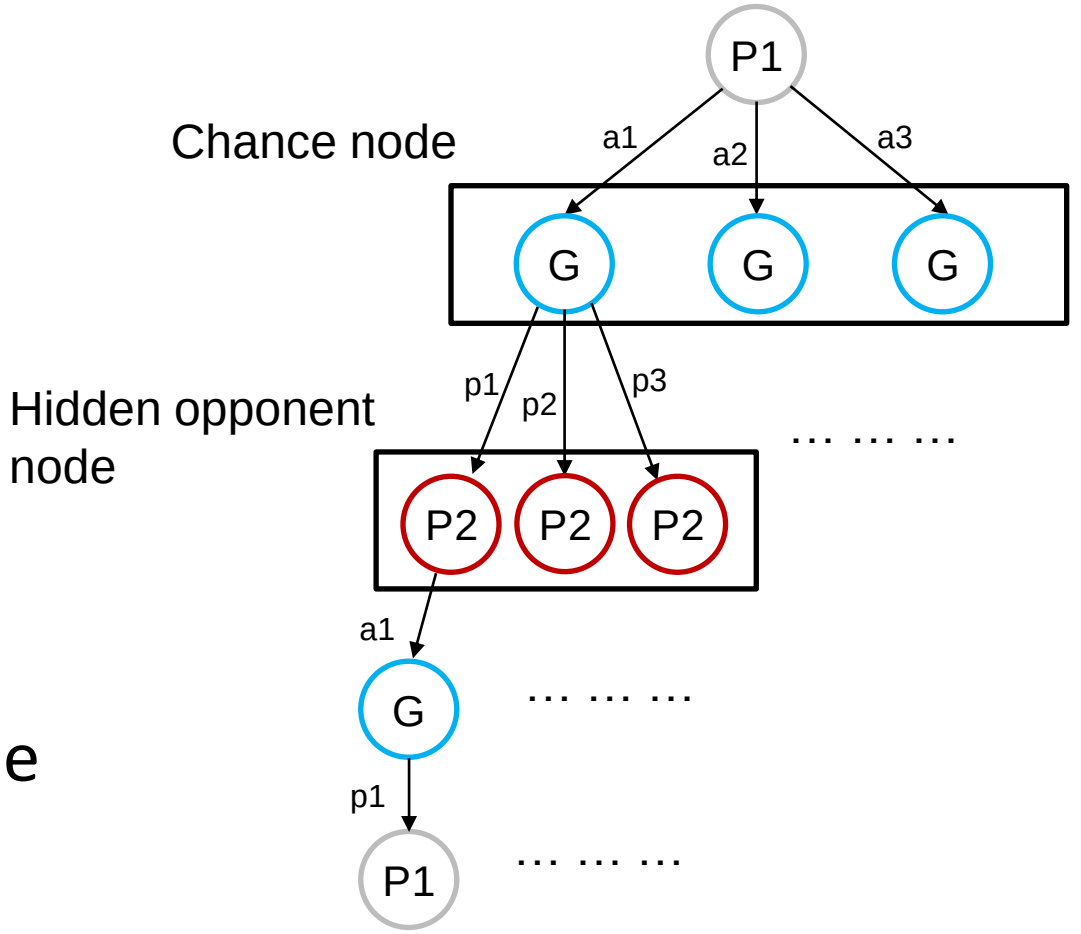
- Random Agent
  - Random select an available action from the action space.
- Reflex Agent
  - Don’t play wild cards, unless they have to.
  - Since the wild cards are least constrained, they would be more useful when there are fewer cards in hand.

DQN Agent

- An improved version of reflex agent.
  - Learn the best reflex policy from history plays.
- Predict the best action from cards in hand.
- DQN Architecture
  - Input: 54*5 binary array: a[i, j] = 1 iff there are j cards of type i in hand.
  - Output: 60*1 array predicting the Q-values for each action.
  - 54*5 – FC512 – FC512 – 60.

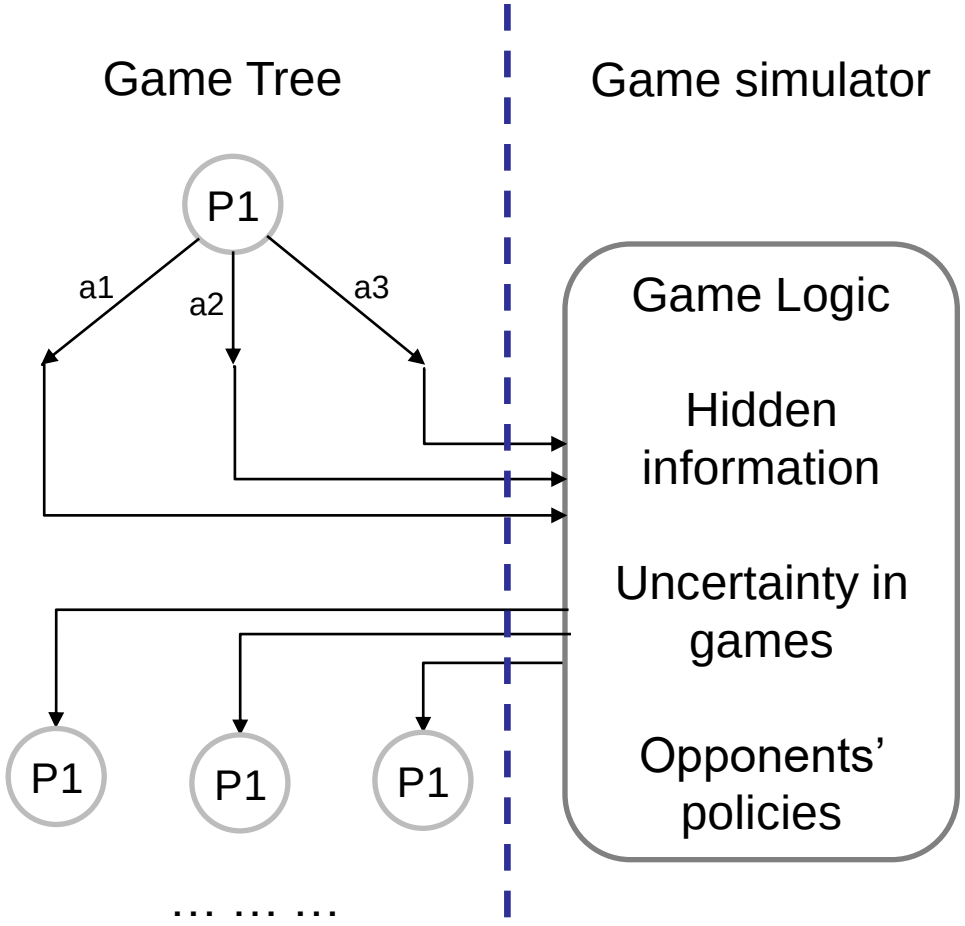
Tree Search Based Methods

- Close loop game tree
  - Include the opponent and stochastic game environment in the game tree
- Not Practical in stochastic imperfect information game
  - Chance nodes and Opponent nodes lead to explosion in tree size
  - Opponent hand information not known



Tree Search Based Methods

- Open loop game tree
  - Hidden states and uncertainty of game excluded from the game tree.
- Useful for simulation-based methods in stochastic imperfect information games
  - Explosion in tree size reduced.
  - Hidden information combined in a good way
  - Similar idea can be used in Expectimax.



Expectimax Agent

- Ignore the hidden information of the opponents
  - Calculate probability of each previous card in the next turn of this agent.
  - This can be solved via Markov Chain under some assumptions.
- The algorithm to calculate the probability of each successor is described as follows:

  Suppose the unseen cards uniformly compose the opponents' hands

  Define $\mathrm{P_h}[i] = \mathrm{P}(\text{card } i \text{ is in hand})$, $\mathrm{P}_j[i] = \mathbf{I}(\text{i can be played after j})$.

  Then $\mathrm{T[j]} = \mathrm{P_h} * \mathrm{P_j}$ (element wise produce)

  If there are k opponents, the probability of each possible previous card is after this agent playing a is $\mathrm{T}^k \mathrm{P}_a$.

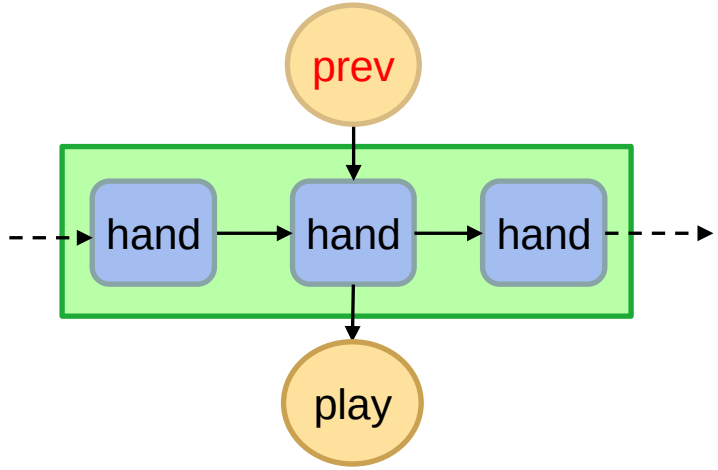
- The rest are the same as Expectimax.

Monte Carlo Tree Search

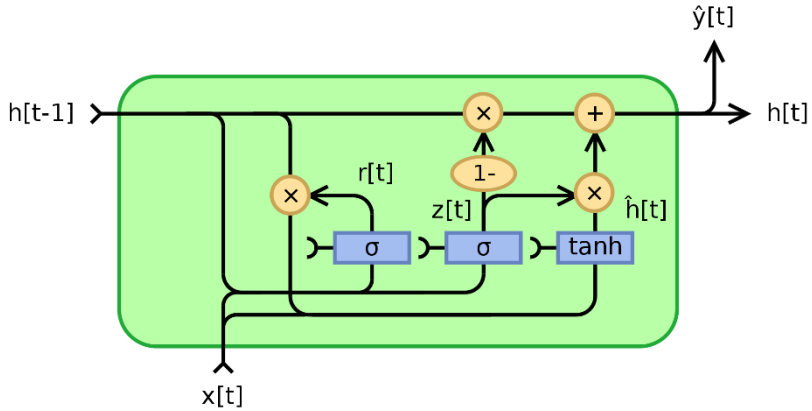
- Monte Carlo tree search is a generic method in game tree search.
- It repeats 4 steps
  - Selection, Expansion, Simulation, Backpropagation
- Vanilla MCTS is not able to solve imperfect games
  - Cannot simulate with hidden states unknown.
- Solution:
  - Predict possible hidden states
  - Predict possible actions.

Dynamic Bayesian Network

- The process of playing cards can be seen as a dynamic Bayesian network.
  - RNNs to predict actions based on the previous card
    - Gated Recurrent Unit
  - Monte Carlo to predict the hidden states
    - i.e., to predict the hands of opponents



Dynamic Bayesian network



Gated Recurrent Unit

Image from https://en.wikipedia.org/wiki/Gated_recurrent_unit

Monte Carlo Based Hand Prediction

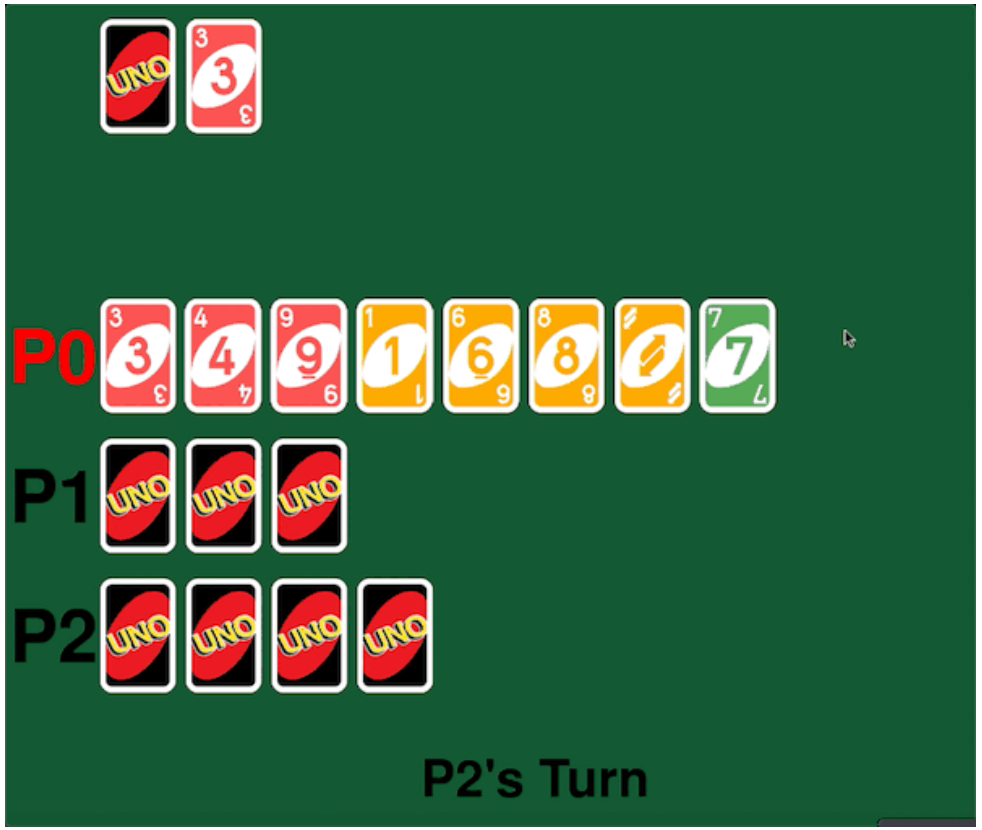
- Idea from likelihood weighting and particle filtering
  - After the main game starts, the current agent starts a number of randomly initialized games.
  - After an opponent has done an action in the main game, require all the agents in the simulated games to do the same action.
    - If one fails to do this action, random perform a valid action and give it weight w_a, where w_a is closer to 0 when more constraints are violated. ($w_a \in (0, 1)$)
    - Otherwise, the weight is set to 1.
  - Resample the games.
- When we need hand of opponents, we can directly sample from the simulated games.

Monte Carlo Tree Search

- Hidden states predicted by the method mentioned above.
- Unclear transition introduced by uncertainty in game.
  - Solution: Open loop
- Simulation can be done now
- Tree search is now available in stochastic imperfect information games.

GUI Results

- GUI based on pygame zero



Reference: https://github.com/bennuttall/uno

Experiments

- Let the agents play with 1, 2 and 3 random agents and record the results.

Number of random agents	1	2	3
[Olivia et al., 2020]	/	0.64	0.46
Reflex	0.95	0.91	0.89
DQN	0.96	0.93	0.91
Expectimax	0.96	0.94	0.92
MCTS	**0.96**	0.95	**0.94**

Table 2: Win rate against random agents (1000 rounds)

Experiments

- Let the agents play against each other and record the results.

	MCTS	DQN	Reflex	Expectimax	Random
MCTS	/	0.55	0.60	0.47	0.98
DQN	0.45	/	0.56	0.48	0.96
Reflex	0.40	0.44	/	0.47	0.95
Expectimax	0.53	0.52	0.53	/	0.96
Random	0.02	0.04	0.05	0.04	/

Table 3: Win rate in 1 vs. 1 matches (100 rounds)

Future Work

- Optimize the game logic to enable better performance in simulations.
- Exploit more domain specific knowledge.

State Isomorphism

- In Go, the states contain exactly the same information if they are symmetric (isomorphic) to each other.
  - Does there exist a similar symmetry in UNO?
  - Yes
- Only consider hand of current player is considered, the followings are some examples of the isomorphic pairs

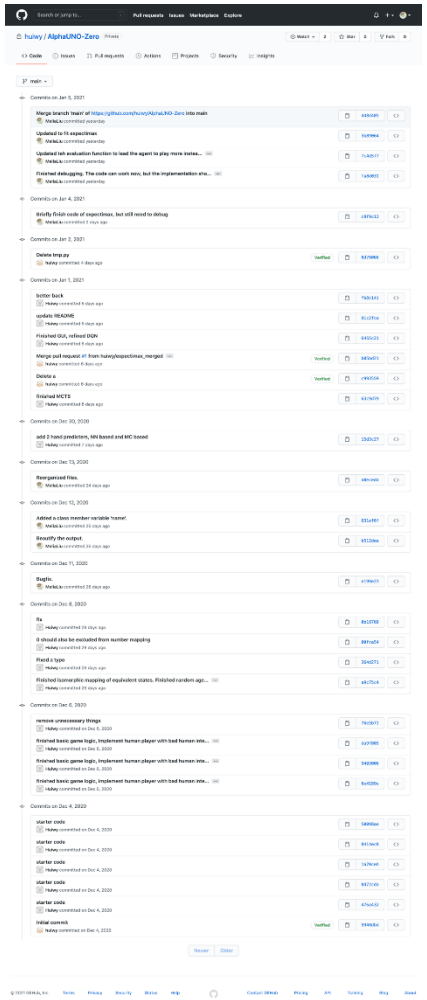
State Isomorphism

- Why state isomorphism?
  - The state space will decrease
  - Efficiency of sampling will increase significantly.

- We have implemented this successfully. However, when game history is considered, more work should be done…

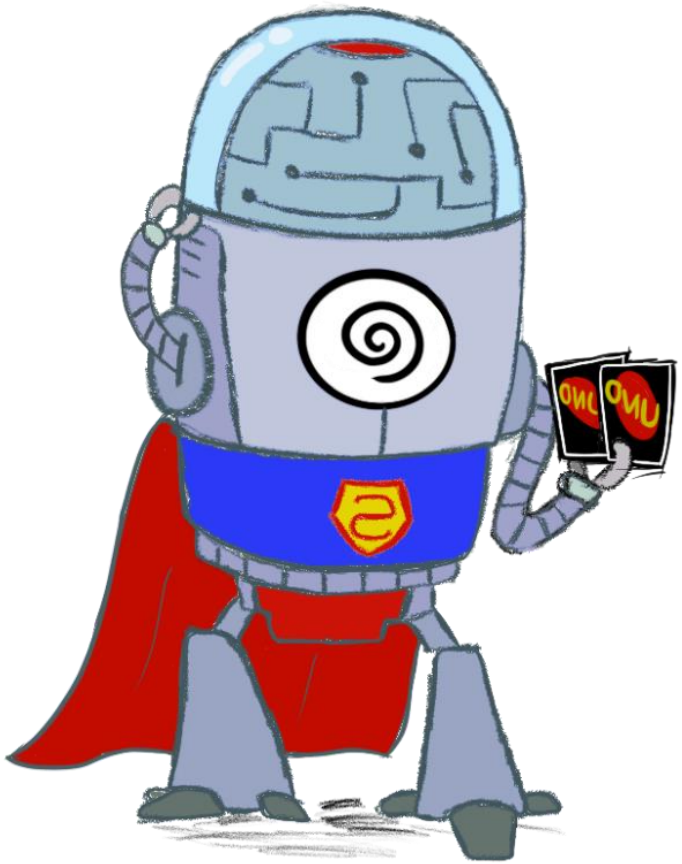
Reproducibility Statement

- We implemented the entire project by ourselves. This is a brief overview of our project. The implementation details are in the report.
- Total loC : 1945.
- Commit history (31 commits).
- The practical impact of our MCTS-HMM agent is a testament to the effectiveness of open science, open source, and open discourse. We will make our code and experimental data publicly available to accommodate reproducibility.



Reference

- Implementation of vanilla MCTS:
  - GeeksforGeeks MCTS
  - Wikipedia MCTS
- Idea of open loop
  - Open Loop Search for General Video Game Playing
- The RNN architecture used in Dynamic Bayesian Network
  - Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling
- Idea of state isomorphism
  - Mastering the game of Go without Human Knowledge
- Other related works
  - Winning Uno With Reinforcement Learning
  - RLCard: A Toolkit for Reinforcement Learning in Card Games



THANKS