4th International Conference on
Computing, Communication, and Intelligent Systems ICCCIS-2023
3-4 November, 2023

Paper ID: 584

Convergence to Nash Equilibrium

A Comparative Study of Game Theory and Reinforcement Learning Algorithms for Rock-Paper-Scissors

Presenter: Geetika Vadali

Index

Problem Statement

To evaluate the performance of **primitive game theory algorithms and evolved reinforcement learning algorithms** on small state-space Imperfect Information Games like Rock-Paper-Scissors for convergence at Nash Equilibrium.

The objective is to assess the five algorithms - Monte Carlo Tree Search, Counterfactual Regret Minimization, Q-Learning, Deep Q-Network and Proximal Policy Optimization based on metrics such as rate of convergence, draw ratio and learning curves.

A 2-agent framework for convergence testing is employed with a standard reward and rule system.

IIG = Imperfect Information Game
RPS = Rock-Paper-Scissors

Existing Solutions

- In imperfect information games, Poker has received drastically more coverage than others, because of its large state and action space. Rock-paper-scissors has comparatively less state space, but because of its cyclicity, it exhibits a resourceful Pareto-optimal zero-sum situation which being easier to interpret, improves readability of the algorithms that are used on it.

- Wang et al. (2014) and Yang et al. (2015) both have utilized the Rock-Paper-Scissors model to study behaviour in competition and in an dynamic environment.

- It was proved way back in 1979 by Khachiyan that the calculation of Nash equilibrium of any game is computationally tractable. This motivated us to regard Nash equilibrium as a state that can only be converged upon.

Existing Solutions

- Methodologies that have been used to train artificial models to play optimally and to gain maximum pay-off have been :

1. **Neural Fictitious Self-Play (2016)** : It works on choosing the best responses against an average behaviour of opponent. A neural network function approximation is added to more generalize the strategy so that it's not always greedy.
2. **ReBeL (2020)** : is a framework of models that combine both deep reinforcement learning and search heuristics in solving imperfect-information games.
3. **Opponent Modelling (2011) :** modelling opponent's strategy in choosing own strategy that maximises pay-off

Existing Solutions absorbed in our research

1. **Monte Carlo Tree Search (2006) -** Combine traditional tree-heuristic with reinforcement learning and it is the algorithm behind DeepMind's AlphaGo.
2. **Counterfactual Regret Minimization (2007) - Zinkevich et al. -** It has been tested on a variant of poker with a very large state space and has computed an approximate equilibrium for the same.

3. **Deep Q-Network (2014) - Google DeepMind -** Q-learning with deep neural networks, introduced to learn and play the Atari 2600 games mastering the complex control policies and strategies. Achieved state-of-the-art results
4. **Proximal Policy Optimization (2017) - Schulman et al. -** In Leduc poker, win rate by the PPO agent is significantly better than human players. In Go, the agent reached an accuracy equivalent to a professional human Go player.

Proposed Work

1. **General Framework for Convergence Testing-** Aim to evaluate the convergence of a 2 player rock paper scissors game at Nash Equilibrium. The agents learn from the environment based on a specified reward scheme.
2. **Reward Scheme-** Agent 1 is rewarded +1, when it wins the game and Agent 2 is rewarded -1, when it wins. In case of draw, both get 0.
3. **Simulations and Iterations-** The algorithms updates and optimizes the policy function in each iterations. Two experiments were conducted, wherein 500 and 1000 simulations were generated.
4. **Termination Condition-** A point where neither of the two agents can earn a better payoff.
5. **Evaluation Metrics-**
    - ❑ Learning Curves
    - ❑ Draw Ratios
    - ❑ Convergence rate, no. of iterations required to reach nash equilibrium.

Proposed Work

6. Algorithms Implemented- The following five algorithms were implemented to update the strategies of the agents-

- **Monte Carlo Tree Search**- It simulates possible outcomes and builds a tree of possible moves and associated values.
- **Counterfactual Regret Minimization**- Iteratively adjusts the strategies to minimize the regret from previous decisions.
- **Q Learning** - Estimates action values in the environment.
- **DQN**- Q-learning is a traditional RL algorithm, while DQN incorporates deep neural networks for improved performance.

  Both Q Learning and DQN are adapted to imperfect information games by modeling the game state as a practically observable markov decision process.

- **PPO**- It optimizes the policy through iteratively updating and sampling from the policy distribution while maintaining a constraint on policy changes.

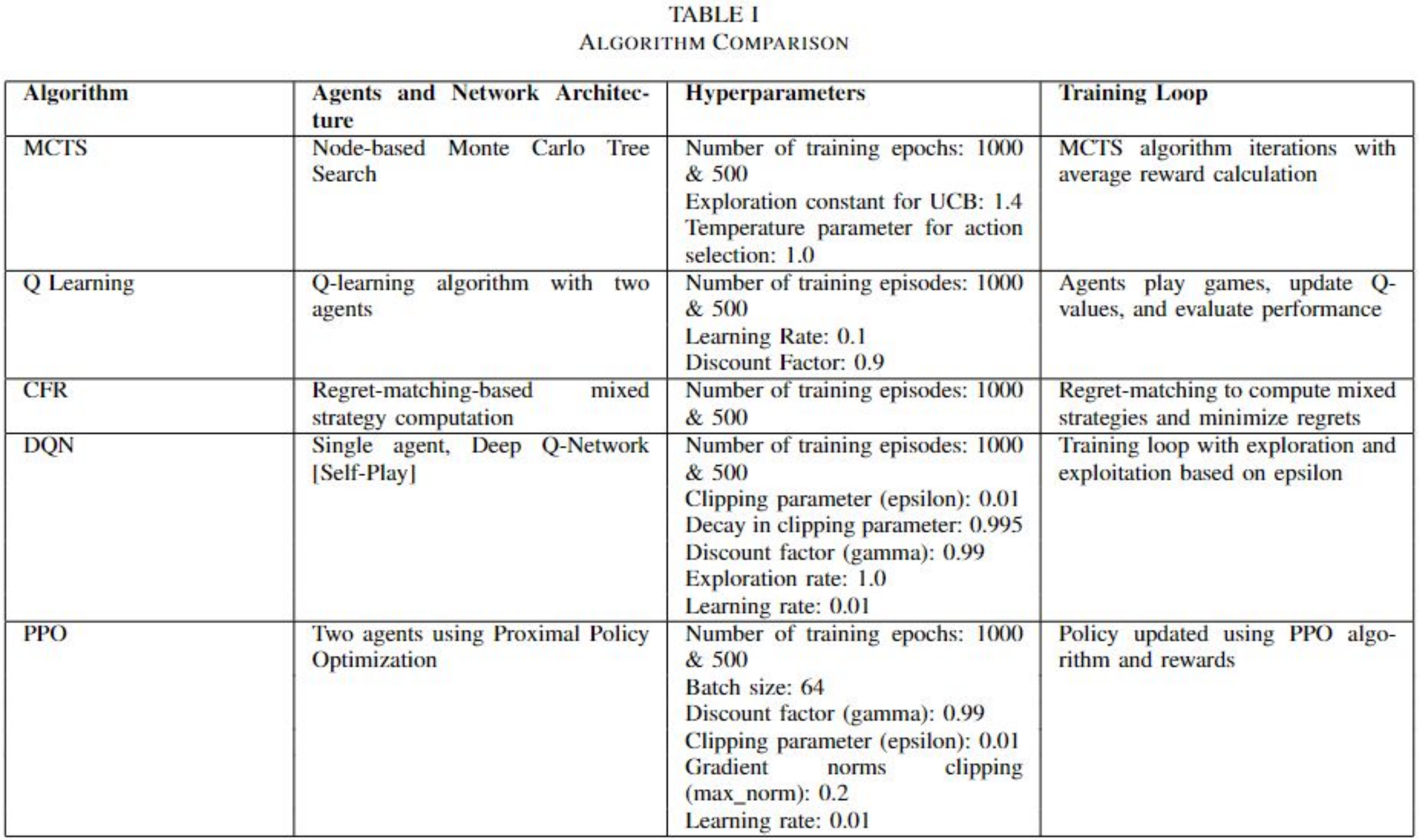
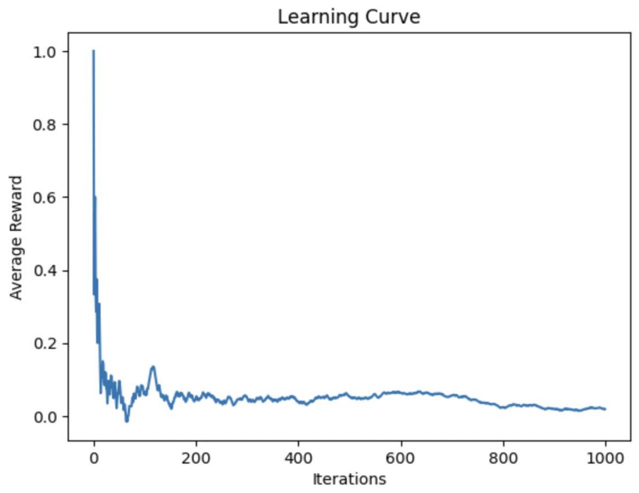


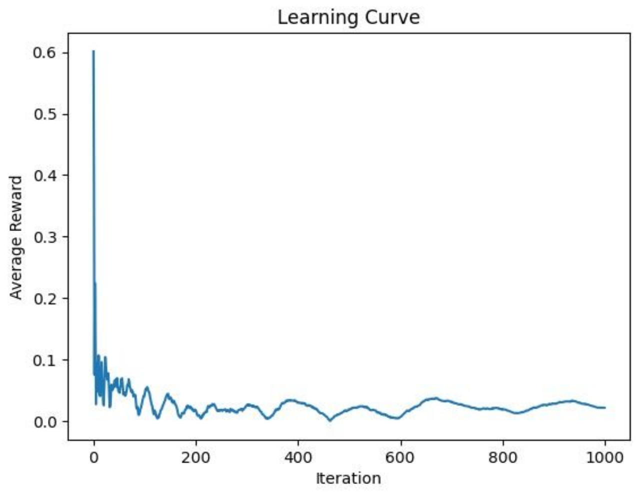
TABLE I
ALGORITHM COMPARISON

Algorithm	Agents and Network Architecture	Hyperparameters	Training Loop
MCTS	Node-based Monte Carlo Tree Search	Number of training epochs: 1000 & 500 Exploration constant for UCB: 1.4 Temperature parameter for action selection: 1.0	MCTS algorithm iterations with average reward calculation
Q Learning	Q-learning algorithm with two agents	Number of training episodes: 1000 & 500 Learning Rate: 0.1 Discount Factor: 0.9	Agents play games, update Q-values, and evaluate performance
CFR	Regret-matching-based mixed strategy computation	Number of training episodes: 1000 & 500	Regret-matching to compute mixed strategies and minimize regrets
DQN	Single agent, Deep Q-Network [Self-Play]	Number of training episodes: 1000 & 500 Clipping parameter (epsilon): 0.01 Decay in clipping parameter: 0.995 Discount factor (gamma): 0.99 Exploration rate: 1.0 Learning rate: 0.01	Training loop with exploration and exploitation based on epsilon
PPO	Two agents using Proximal Policy Optimization	Number of training epochs: 1000 & 500 Batch size: 64 Discount factor (gamma): 0.99 Clipping parameter (epsilon): 0.01 Gradient norms clipping (max_norm): 0.2 Learning rate: 0.01	Policy updated using PPO algorithm and rewards

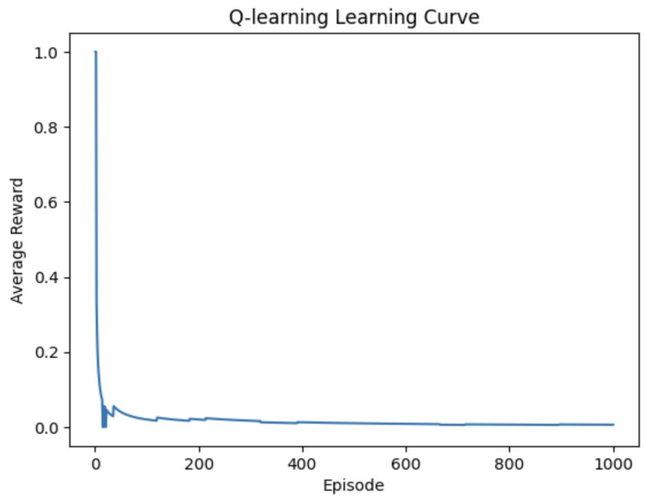
Results - Learning Curves for 1000 iterations



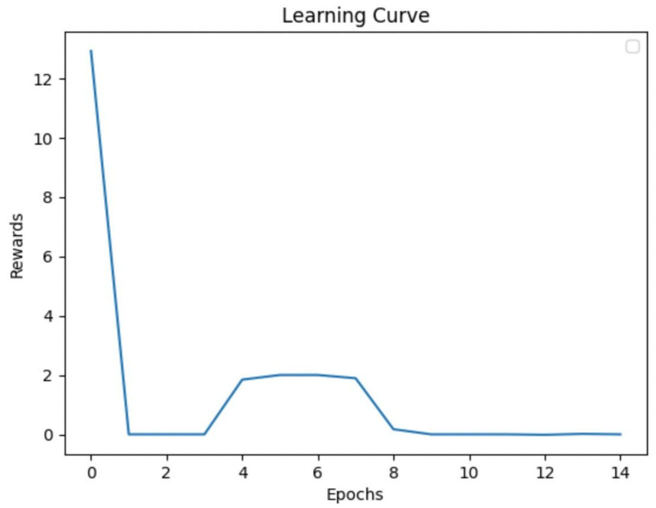
MCTS



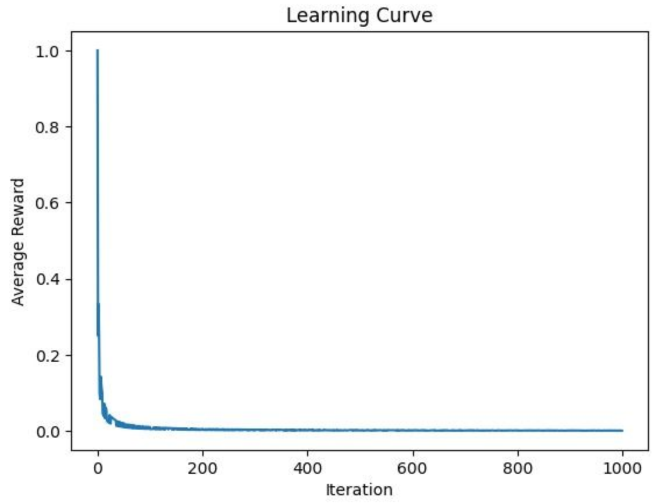
CFR



Q-Learning



PPO



DQN

Results - Draw Ratio for 1000 iterations

Algorithm	Draw Ratio
MCTS	0.341
CFR	0.991
Q-Learning	0.666
PPO	0.356
DQN	0.490

Draw Ratio = Number of Draws / Total number of Iterations

More the number of draws => better convergence to Nash equilibrium.
As the average reward leans towards 0.0, draw state is more commonly encountered and that represents the equilibrium state of game play and the agent having no incentive to change strategy.

Concluding the collective results:

- **DQN** converges very early on to 0.0 average reward and yet does not have equivalently large draw ratio.
- **CFR** has very large draw ratio, but based off the learning curve, CFR is suspected to explore more than exploit.
- **Q-Learning** is concluded to be efficient according to convergence to equilibrium reward as well as the draw ratio. Q-learning is found to have reached the most stable balance to explore and exploit.

Discussion and Future Scope

- We believe that the results as collected for the RPS game might be transferable to any **small state-action-space IIG**. But most real-world multi-agent situations have extremely **large parameter set** and state space where it becomes even more necessary for the machine to be extremely accurate.
- DQN and PPO have been proven to work very well for these large information spaces and hence their capability cannot be boiled down to their performance on RPS. And that implies, the comparative analysis needs to be conducted on larger games too like Poker.
- We want to explore other equilibrium strategies like Pareto Optimal Equilibrium to understand zero-sum IIGs.
- Exploring imperfect information game theory is vital to understand the similarly complex multi-agent and multi-state dynamics of real world. Getting to very efficient equilibrium strategies will help in **advancing machine intelligence** and **decision-making**.

Thank You and open to questions!