Convergence to Nash Equilibrium: A Comparative Study of Rock-Paper-Scissors Algorithms

Geetika Vadali
Department of AI and DS
Indira Gandhi Delhi Technical University for Women
Delhi, India
geetika.vadali4@gmail.com

M Deekshitha Reddy
Department of AI and DS
Indira Gandhi Delhi Technical University for Women
Delhi, India
deekshitha.m.reddy@gmail.com

Ritu Rani
Center of Excellence-Artificial Intelligence
Indira Gandhi Delhi Technical University for Women
Delhi, India
riturani@igdtuw.ac.in

Poonam Bansal
Department of AI and DS
Indira Gandhi Delhi Technical University for Women
Delhi, India
poonambansal@igdtuw.ac.in

***Abstract*—Rock-paper-scissors is one of the most established imperfect information games in Game theory. The Nash Equilibrium of an RPS game is relatively simple but computationally intractable; hence, various algorithms are employed to converge to a state of maximum payoff. In this paper, five algorithms, namely - Counterfactual Regret Minimization, Monte Carlo Tree Search, Q-learning, Deep Q-Network and Proximal Policy Optimization, have been compared on the evaluation metrics of average reward, draw ratio and convergence speed. Throughout the comparative analysis, visualising the learning curves, and qualitative comparison, Q-learning has shown the best convergence to Nash equilibrium for RPS.**

***Index Terms*—Game Theory, Imperfect Information Game, Rock-Paper-Scissors, Reinforcement Learning, Regret, Nash Equilibrium, Deep Reinforcement learning**

I. Introduction

A. Fundamentals of Game Theory

Game theory studies the strategic behaviour of more than one rational agent. These agents are called players. Players participate in the game; they have available some actions that they can play. All the players make their moves simultaneously. And the payoff is the resulting "value" or "utility" of the actions taken. A significant question in game theory is determining the strategy each player should use to maximise their payoff from the game.

Even in strategies, there are two kinds of strategies that can be taken up by a player – pure strategy, where the player utilising pure strategy always chooses the same action, e.g. a player always throwing a scissors hand in rock-paper-scissors, or a player always choosing to challenge in poker. In this strategy, simple combinatorics can lead us to the payoff the player can expect. In contrast, there is another strategy scheme called mixed strategy, where the player chooses an action based on a probability distribution over all the available options. Here, determining the best/optimal strategy for a player-facing mixed strategy is not straightforward.

Games played by players can be classified into zero-sum and general-sum based on the relation between the payoff of one player concerning their opponent. In two-person zero-sum games, if α denotes the payoff to player one and β denotes the payoff to player 2, then α = - β everywhere through the payoff matrix. In general-sum, there is no particular restriction between these two (or multiple, in case of multiple agents) values.

A well-known example where both players have three available actions is rock-paper-scissors. The matrix here looks like this:

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

In an iterated version where the action to be played next is chosen after the opponent plays their move, players must adopt a pure strategy to win. This is because the other player can always ensure a winning payoff of "1" after learning the current player's unwavering strategy. However, both players can still maximize their payoff when they start using a mixed strategy of probabilistically choosing their actions over iterations.

B. Imperfect Information Games

A normal-form game is defined as the matrix representation of the utility of every player for every combination of available actions in a system where the states of the world depend fully on the cumulative of player's actions. Normal-form representation though proves to be impractical because most real-world games evolve in time increasing the number of strategies to randomize over. Hence these games then use an extensive form representation, which in general is exponentially smaller and it does not make the assumption that the players will always play simultaneously. While the structure of representing games in normal-form was a matrix, in extensive form, they are represented using trees, where the nodes of the tree represent

ISBN: 979-8-3503-0611-8/23/$31.00 ©2023 IEEE

Games, in general can be bifurcated into perfect-information games and imperfect-information games based on the existing knowledge of one player about the other's strategy and policy. The players have complete knowledge of the game structure and the sequence in which the opponent plays the moves. Perfect information games that occur in turns assume that the players have the observation of the state of the game as well as the previously played moves before playing their own turn. This assumption is usually impractical in designing dynamic reactive systems because most of them have a latent space invisible to the active players. Since the execution of all of these components are simultaneous, each player chooses actions independently of the other components. These situations call for defining some games as imperfect information where the system components only have partial information about the game, state, play and actions.

C. Nash Equilibrium

A Nash equilibrium is defined to be a stable strategy profile where no player would want to consciously change her strategy unless she knew what strategies the other players were implementing. In games with two or more players, if each player has chosen a strategy and no one is expected to elevate one's payoff by changing their strategy only while other players are not supposed to have an unchanging strategy, then the current set of strategy choices, sequence of actions makes up the Nash Equilibrium. The best response is defined as the strategy in game theory that generates the most favourable utility outcome for a player, taking into account the other players' strategies.

Nash equilibrium is thereby also defined as the point at which all players in a game have selected the best response with respect to the other players.

D. Rock, Paper, Scissors

Rock-paper-scissors is one of the integral non-cooperative two-player zero-sum imperfect-information-game. Rock-paper-scissors, which will hereby be abbreviated as RPS, projects a model for assessing contention caused by cyclic dominance [1]. RPS consists of three possible action states - R, P and S, where R is for rock, P is for paper and S is for scissors. Action R beats S, which itself beats P, which in turn beats R. This essentially means none of the three actions is a fool-proof winning choice. Classical game theory gives importance to the players' rationality; they want not only maximisation of their own payoff but also minimisation of the adversary's. Since repetition of choices might be easily identified and exploited by the rational and intelligent opponent, it is only safe for the player to choose each iteration of the game independent of what actions have been chosen thus far.

With a mixed strategy where all probabilities are positive, it will be impossible for player B to exploit player A if the latter completely randomises their choice of actions.

Setting the mixed strategy to be (1/3, 1/3, 1/3) is seen to reach the most stable point in RPS play, diverting from which no player can expect any better pay-off. Hence by definition, this is where the Nash Equilibrium of RPS lies.

There has been extensive research and implementation of equilibrium searching algorithms on Poker and other multi-agent games with massive information set sizes. In this paper, the cyclicity of Rock-Paper-Scissors is studied in detail while analysing the convergence of various state-of-the-art algorithms toward the unique Nash Equilibrium.

II. Literature Review

Rock-Paper-Scissors is a non-cooperative two-player zero-sum game of cyclic dominance. The game constitutes non-randomness, best played in rapid rhythm and originates from the ancient Japanese concept of Sansukumi, where three forces keep each other in check. According to Wang et al. [2], the RPS game has been a basic model system to study the behaviour and decision-making of human subjects in competitive environments and the corresponding social dynamics. The same is pronounced by Yang et al. [3].

Nash Equilibrium for any game is computationally tractable [4] and even before that, along with the advent of game theory in economics, theorised that any two-person zero-sum game must have an equilibrium, a state from which none of the player would want to deviate to ensure themselves maximum pay-off. [5]

Heinrich et al. [6] introduced Neural Fictitious Self-play that extends on Fictitious Self-play by the same authors in 2015. This mythology works on choosing the best responses i.e., optimal strategies against the opponent's average behaviour in extensive-form games added to which is a neural network function approximation. Brown et al. [7] combined deep reinforcement learning with search heuristics in solving imperfect-information games. They introduced ReBeL, a framework that reduces to AlphaZero for perfect-information games, but for imperfect-information games like RPS and Poker, it approximately converges to the Nash Equilibrium. Game-theory-based opponent modelling by Tumer et al. [8] developed an algorithm combining pure opponent modelling and game theory to play the best response against the suboptimal opponent.

There has been a growing interest in using RL to play imperfect information games. In particular, PPO is effective in various games, including poker, Go, and StarCraft. For example, in a study by Silver et al. [9], a PPO agent was trained to play Leduc poker. The agent achieved a win rate of over 60%, significantly better than human players. Deep reinforcement learning from self-play in imperfect-information games.

In another study by Brown et al. [10], a PPO agent was trained to play Go. The agent was able to achieve a dan rating of 3, which is equivalent to a professional human player. Mnih et al. [11], Deep Q-Learning with Experience Replay was introduced for input game state images and let the network master the complex control policies and strategies of Atari 2600 computer games. For the seven sub-games it was trained on, DQN enabled state-of-the-art results in six of the seven games.

III. Proposed Methodology

We use a general framework that can be used to test convergence at Nash equilibrium for any two-player Rock-Paper-Scissors game. Fig. 1 outlines the framework used for all the algorithms i.e., two agents and the corresponding reward scheme is initiated in the environment. When agent 1 wins the game, it is rewarded by +1. Conversely, when agent 2 wins the game, it is rewarded by -1. In the next step, the gameplay is simulated for 1000 and 500 iterations in two different experiments. The algorithm updates and optimises the policy function in each iteration. Further, the algorithm is terminated when the two agents reach a point where no one can earn a better payoff. During the process, learning curves, convergence rate, draw ratio and number of iterations needed to reach Nash Equilibrium are recorded for further examination and comparison of the five algorithms.

The five different algorithms used to update the strategies of the agents in this paper are - Monte Carlo Tree Search, Q-Learning RL, Counterfactual Regret Minimization, Deep Q-Network and Proximal Policy Optimization. The next section gives detailed functions employed in each technique.

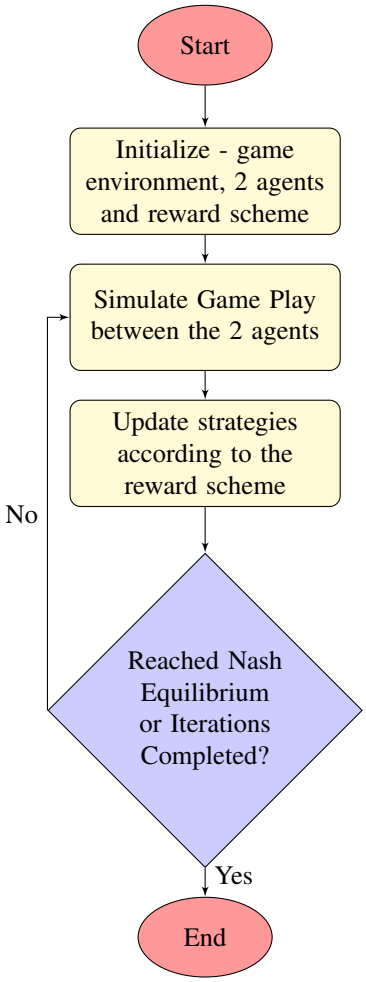

Fig. 1. Process of finding Nash Equilibrium

IV. Model Implementations

A. Monte Carlo Tree Search

Monte Carlo Tree Search (MCTS) is a heuristic algorithm that is employed to solve complex game trees [12]. By iteratively expanding a game tree and evaluating the nodes within it, MCTS combines random simulations with a heuristic function to estimate the value of each node. The process begins with the expansion of the root node, which contains the game's current state. From there, a child node is randomly selected, and a game simulation starts from that node. This simulation continues until the game reaches its conclusion. Then the heuristic value of the corresponding child node is updated with the output of the simulation. The iteration is repeated for all child nodes stemming from the root node. Further expansion is performed on the child node with the highest heuristic value. The process continues till leaf node is reached. Backpropagation is employed to propagate the value of the leaf node to the root node back. The value of the leaf node by the probability of reaching that node. The resulting value of the root node is then used to decide the next action to be picked in the game. Demonstrating remarkable effectiveness, MCTS has achieved cutting-edge results in various board games, including Go [13] [14], Chess, and Shogi.

B. Q-Learning Reinforcement Learning

Q-learning [15] is an off-policy reinforcement learning algorithm. The agent learns from previous experiences without using an environment model and instead uses temporal difference error for value updates. It is a technique to discover an agent's optimal policy in the given environment. The q-values denote the anticipated return of taking a particular action in a particular state. The Q-table values are updated repeatedly using the Bellman equation:

$$Q(s,a) = \mathrm{r} + \gamma * max_{a'}\ Q(s',a')$$

where: s represents the present state, a represents the present action, r is the reward received upon taking a in s, γ is the discount factor, s' represents the next state after taking a in s, a' is the optimal action in state s'.

The Q-values are initialized randomly, the agents pick actions and receive rewards interacting with the environment. After each interaction, the Q-values are updated using the Bellman equation. This process continues until the Q-values converge to the optimal policy.

C. Counterfactual Regret Minimization

Regret is defined as the implication of not having chosen an action that would have guaranteed a better or best payoff for the latest round of gameplay. These positive regrets inform the future decisions of the player while choosing their strategy. Regret may be represented by the difference between the chosen action's utility and the utility of the more optimal

action choice, with respect to the fixed choices of all other players. [16]

$$R_N = \mu^* n - \mu_j \sum_{j=1}^{K} E[T_j(n)]$$

where R_N is the regret of the player after n plays, μ^* is defined as the best possible anticipated reward. The term inside the summation represents the expected amount of plays for a player j in the beginning n trials of the game.

Deriving an example from RPS, [17] if in a certain round player A plays scissors and player B plays scissors too, as player A you would regret not having played rock – thus playing rock obtains a positive regret and considering the opponent sticks to their strategy of throwing scissors, regret matching would earn player A better payoff by playing rock. Whereas, the action of throwing paper would earn negative regret because that will definitely lower the payoff to player A who would lose more than the draw currently. Hence, player A will be de-incentivized to choose paper in the next rounds of game play. CFR is an iterative algorithm that will converge to a Nash equilibrium with a theoretical convergence bound of $O(1/\sqrt{T})$ in any finite two-player zero-sum game. CFR algorithm utilizes the regret matching algorithm. Additionally, it factors in the probabilities of reaching every information set as in the player's strategies and considering the game is iterative or sequential in moving through the information sets, CFR does a passing forward of the information of the game.

D. Deep Q-Network

Deep Q-Network of Deep RL paradigm was proposed by DeepMind in 2015 working on visual Atari games. [11] The algorithm works on the fundamentals of Q-learning, deep neural networks, and experience replay. Q-learning updates the utility, cost and reward system when the agent moves from s to s'. It estimates the policy that the agent can follow to ensure that it can be guaranteed the maximum payoff. In DQN, we try to approximate a complex non-linear function (s,a) using a neural network.

The act of sampling a tiny batch of tuples from the replay buffer to learn the optimal policy is termed experience replay.

The difference between Q-learning and deep Q-network learning lies in the training phase, where instead of updating the state-action pair's Q-value directly, we make use of a loss function that will compare the Q-value predictions with the target and do gradient descent to update the weights of the Q-network and in turn predict better.

$$y_i = r_i + \gamma * max_{a' \in A} Q(s_i^{'}, a_i)$$

where r_i represents the reward for the ith game-play, γ is the discount factor derived from the Bellman equation. a_i is the action at s_i state, $s_i^{'}$ represents the next state. [18]

E. Proximal Policy Optimization

Proximal Policy Optimization (PPO) [19] has emerged as a prominent algorithm in the field of reinforcement learning. A temporal, model-free, on-policy algorithm utilises a surrogate objective function to approximate the true policy gradient. Clipped surrogate objective function:

$$L^{CLIP}(\theta) = \hat{E}_t[min(r_t(\theta)\hat{A}_t, clip(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t]$$

where: θ are the parameters of the policy, $r_t(\theta)$ is the probability of picking a action in a particular state under the current policy, $\hat{A}_t$ is the advantage function, which measures how much better a particular action is than the average action, ϵ controls the amount of clipping $\hat{E}_t$ is the expectation over all trajectories

The PPO algorithm policy is updated by maximising the clipped surrogate objective function. It is achieved using a stochastic gradient descent algorithm. By constraining this surrogate objective function to remain close to the current policy, PPO ensures stability and prevents significant deviations from the current policy state.

Moreover, PPO employs importance sampling techniques to reduce the variance of gradient estimates, resulting in increased efficiency compared to other policy gradient methods. PPO has been a very constructive reinforcement learning algorithm. It has been used to achieve cutting-edge results on various tasks, including Atari games, robotics, and continuous control.

V. Results and Discussion

In this experiment we compared the five algorithms based on a specific reward scheme. Agent 1 was assigned a reward of +1 for winning a game, while agent 2 was assigned -1 for winning a game. We monitored the behaviour of the five algorithms based on the convergence and draw ratios until they reached Nash Equilibrium. To compare the convergence, we initialized agent 1 to win in the first stage.

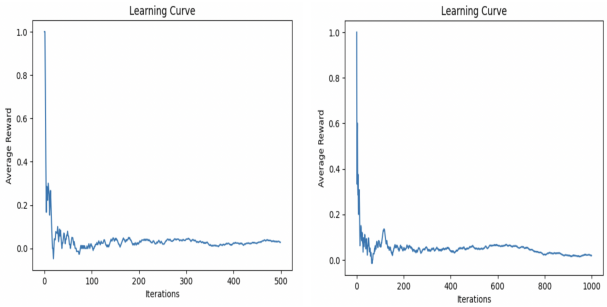

Fig. 2. MCTS Algorithm: 500 and 1000 Simulations

Practical analysis of the learning curves directly relates to the average reward that algorithms expect throughout the training. Average reward is the sum of all rewards calculated per episode i.e., 1 stands for win, 0 for a draw and -1 for a loss, divided by the number of iterations (500 and 1000). Based

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

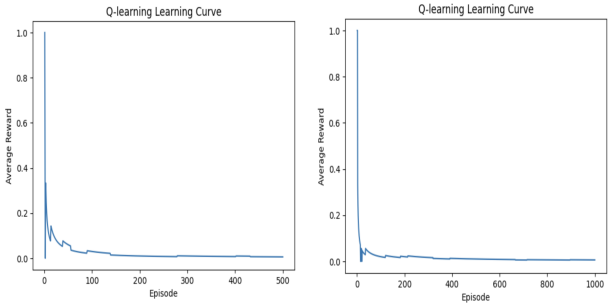

Fig. 3. Q Learning Algorithm: 500 and 1000 Simulations

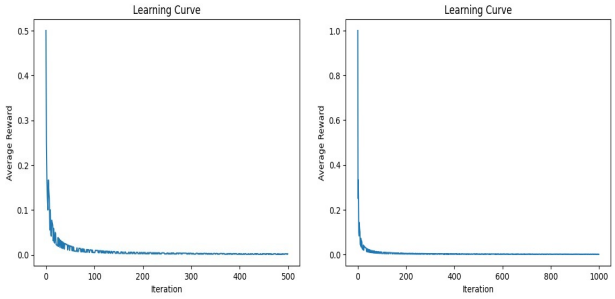

Fig. 5. DQN Algorithm: 500 and 1000 Simulations

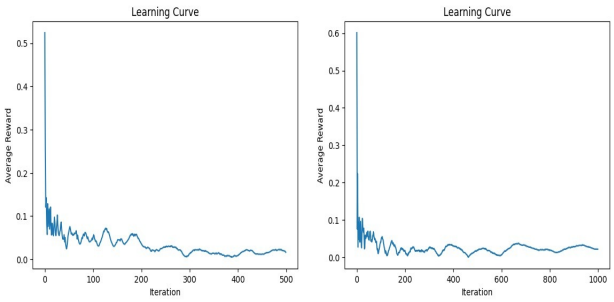

Fig. 4. CFR Algorithm: 500 and 1000 Simulations

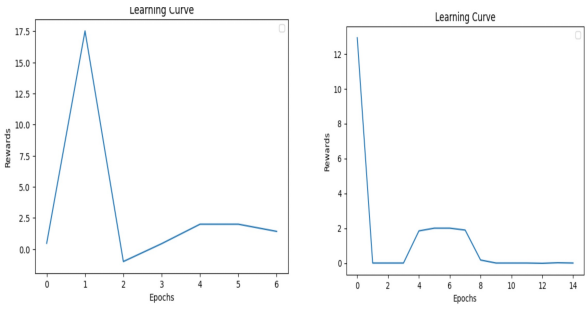

Fig. 6. PPO Algorithm: 500 and 1000 Simulations

on the average reward's convergence to 0.0, agents tend to draw more when they have learnt the strategies for maximising their payoff and minimising the opponents'. Hence, the rate at which the average reward for any algorithm converges to 0.0 determines the rate of convergence of the same towards Nash Equilibrium. Thereby, DQN and Q-learning algorithms are observed to have the earliest convergences to Nash equilibrium. These are followed by the MCTS which being a non-RL algorithm has better understanding of the less states of the RPS game. Although it is observed that it is fairly turbulent

TABLE II
ALGORITHM CONVERGENCE IN ITERATIONS WITH THE RESPECT TO DRAW RATIO

Algorithm	Episodes	Draw Ratio
MCTS	500	0.354
	1000	0.341
Q-Learning	500	0.926
	1000	0.991
CFR	500	0.667
	1000	0.666
DQN	500	0.482
	1000	0.490
PPO	500	0.917
	1000	0.356

at the beginning and gets stable slowly, this might be because it is exploring more than exploiting the game strategies. CFR is turbulent throughout the learning period, pointing to the hypothesis of its exploration tendency. PPO on the other hand was observed to be disparate between the the two learning lengths.

As indicated in table 2, Q-Learning and CFR algorithms showed the closest approximation to the Nash equilibrium. Specifically, Q-Learning achieved a draw ratio of 0.991 with 1000 episodes, while CFR achieved draw ratio of 0.666 for 1000 episodes. On the other hand, with MCTS having the lowest draw ratio, the Exploration-Exploitation system is imbalanced. MCTS agents were not able to exploit the high-rewarding action set. The performance of PPO was very sensitive to training episodes. The crucial difference in draw ratio between 500 (0.917) and 1000 (0.356) episodes could indicate that the algorithm did not learn the optimal policy efficiently for the latter case. Insufficient exploration of suboptimal policies might have contributed to this discrepancy. The DQN algorithm gives a draw ratio between 0.482-0.490 despite early convergence with equal explorative and exploitative learning employed by the agent, which could be accounted to the information set size. DQN algorithm has proved its efficiency for games like Atari with larger action set. [11]

Based on convergence rate and draw ratio, Q-Learning is the optimal algorithm for Rock-Paper-Scissors.

VI. CONCLUSION AND FUTURE SCOPE

Several avenues for future work can be explored to further enhance our understanding of the five algorithms studied in this paper. Firstly, evaluating the algorithms in more complex and real-world environments, such as the stock market, which has a more diverse reward scheme and more agents. Investigating alternative equilibrium concepts, such as Pareto optimal equilibria, will illuminate the algorithms' capabilities in games where a Nash equilibrium is unclear. Moreover, extending the analysis to multiplayer games will enable the examination of how the algorithms perform in settings with multiple agents, allowing for a deeper understanding of their effectiveness in complex, multi-agent interactions. Furthermore, exploring the algorithms' performance in various game environments, including those with imperfect information or sequential decision-making, will comprehensively evaluate their adaptability and robustness. By pursuing these avenues, we can advance our knowledge of algorithms for imperfect information games and their applicability in real-world scenarios.

REFERENCES

[1] A. Szolnoki, M. Mobilia, L.-L. Jiang, B. Szczesny, A. M. Rucklidge, and M. Perc, "Cyclic dominance in evolutionary games: a review," *Journal of The Royal Society Interface*, vol. 11, p. 20140735, Nov. 2014.

[2] Z. Wang, B. Xu, and H.-J. Zhou, "Social cycling and conditional responses in the rock-paper-scissors game," *Scientific Reports*, vol. 4, July 2014.

[3] X. Yang, H. Zhou, and X. Zhou, "Rock paper scissors: CRISPR/cas9-mediated interference with geminiviruses in plants," *Science China Life Sciences*, vol. 62, pp. 1389–1391, Sept. 2019.

[4] L. G. Khachiyan, ""a polynomial algorithm in linear programming,"," 1979.

[5] J. von Neumann, ""zur theorie der gesellschaftsspiele"," 1928.

[6] J. Heinrich and D. Silver, "Deep reinforcement learning from self-play in imperfect-information games," 2016.

[7] N. Brown, A. Bakhtin, A. Lerer, and Q. Gong, "Combining deep reinforcement learning and search for imperfect-information games," 2020.

[8] 10th Int. Conf. on Autonomous Agents, Y. S. Multiagent Systems (AAMAS 2011), Tumer, and S. (eds.), "Game theory-based opponent modeling in large imperfectinformation games, sam ganzfried and tuomas sandholm, proc. of," May, 2–6, 2011, Taipei, Taiwan, pp. XXX-XXX.

[9] "Mastering the game of go with deep neural networks and tree search," 2016.

[10] N. Brown, A. Lerer, S. Gross, and T. Sandholm, "Deep counterfactual regret minimization," 2019.

[11] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," 2013.

[12] G. M. J.-B. C. Chaslot, *Monte-Carlo Tree Search*. PhD thesis.

[13] S. Gelly and D. Silver, "Achieving master level play in 9 x 9 computer go.," in *AAAI*, vol. 8, pp. 1537–1540, 2008.

[14] S. Gelly, L. Kocsis, M. Schoenauer, M. Sebag, D. Silver, C. Szepesvári, and O. Teytaud, "The grand challenge of computer go: Monte carlo tree search and extensions," *Commun. ACM*, vol. 55, p. 106–113, mar 2012.

[15] C. J. C. H. Watkins, "Learning from delayed rewards," 1989.

[16] M. Zinkevich, M. Johanson, M. Bowling, and C. Piccione, "Regret minimization in games with incomplete information," *Advances in neural information processing systems*, vol. 20, 2007.

[17] T. W. Neller and M. Lanctot, "An introduction to counterfactual regret minimization," in *Proceedings of Model AI Assignments, The Fourth Symposium on Educational Advances in Artificial Intelligence (EAAI-2013)*, vol. 11, 2013.

[18] J. Fan, Z. Wang, Y. Xie, and Z. Yang, "A theoretical analysis of deep q-learning," in *Proceedings of the 2nd Conference on Learning for Dynamics and Control* (A. M. Bayen, A. Jadbabaie, G. Pappas, P. A. Parrilo, B. Recht, C. Tomlin, and M. Zeilinger, eds.), vol. 120 of *Proceedings of Machine Learning Research*, pp. 486–489, PMLR, 10–11 Jun 2020.

[19] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017.