Rational and Convergent Learning in Stochastic Games

Reproduced by: Alexis Francois Verdoodt[1], Lennert Bontinck[1], Sofyan Ajridi[1] and Wolf De Wulf[1]

Vrije Universiteit Brussel[1]
(firstname.middlename.lastname@vub.be)

Abstract

This article contains a reproduction and extension of the experiments in the paper from Bowling and Veloso on rationality and convergence in stochastic games (2001). The algorithm that is introduced in this paper is briefly explained and the experiments that were devised for evaluating its performance are re-enacted. Originally, the experiments were restricted to self-play environments. As an extension to the analysis, experiments in non-self-play environments are presented.

1 Introduction

Two desirable properties that multiagent learning algorithms can have are *rationality* and *convergence*. An algorithm that is rational is one that learns the best-response with respect to the actions of the other agents. On the other hand, an algorithm that is convergent is one that finds a stationary policy. In general, it is desirable for learning algorithms to have both. There are lots of different multiagent learning algorithms that each have their own way of trying to achieve at least one of the two properties, for example JALs (Claus and Boutilier, 1998), minimax-Q (Littman, 1994a), Nash Q (Hu and Wellman, 2003) or Regret Matching (Hart and Mas-Colell, 2000). Bowling and Veloso state that none of them have been able to achieve both and they present their WoLF algorithm for which they prove that it is both rational and convergent. The algorithm achieves convergence by extending the already rational policy hill-climbing algorithm with a variable learning rate using the "Win or Lose Fast" principle. The stochastic game framework (Shapley, 1953) is used to examine the WoLF algorithm's performance. The original experiments were restricted to 2-player games in self-play environments, which means that both agents always implement the same algorithm to learn. Take note that Bowling and Veloso's work has aged and at this time there are already algorithms for which it is proven that they have both the rationality and convergence properties for games with an arbitrary number of players and actions (e.g. AWESOME (Conitzer and Sandholm, 2003)). The extension to Bowling and Veloso's work that is provided in this article is the analysis of the convergence and rationality of the WoLF algorithm in 2-player games where the two agents do *not* implement the same learning strategy.

Note that rudimentary game theory concepts are not addressed, however knowledge of them is required for understanding the analysis. The focus lies on reproducing and extending the original experiments. Section 2 explains the policy hill climbing algorithm and clarifies how the WoLF algorithm builds on it. In Section 3, a brief explanation of the four games that were used in the original experiments is given. This section also explains how the four games were originally simulated and how the simulations were extended. Intertwined with these explanations are the results. The article concludes in section 4 with a discussion on the findings.

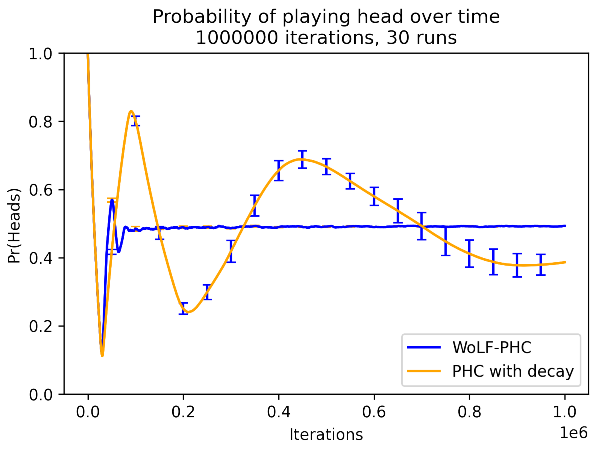
2 Methods

Bowling and Veloso's formal definitions for the rationality and convergence properties are given below. Knowing what these properties comprise is needed for understanding the analysis.

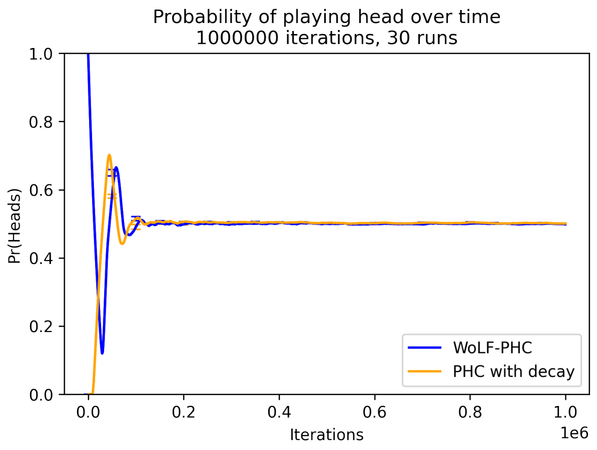
Rationality *If the other players' policies converge to stationary policies then the learning algorithm will converge to a policy that is a best-response to their policies*

Convergence *The learner will necessarily converge to a stationary policy. This property will usually be conditioned on the other agents using an algorithm from some class of learning algorithms*

In general, most 'useful' algorithms have the rationality property. An algorithm that cannot react optimally to other players playing a stationary strategy would not be that useful. Convergence is a different matter. The definition for the convergence property is always used with respect to some class of other learning algorithms. Bowling and Veloso chose to only analyse the convergence property for their WoLF algorithm in the case of self-play. As mentioned before, this article extends their analysis by looking



(a) Self-play



(b) One WoLF-PHC agent, one PHC agent

Figure 1: Average policy for playing head in the matching pennies game. The error bars indicate the standard error. Initial policies were fixed to one player starting with probability 1 for playing heads while the other player started with probability 1 for playing tails. For this game, the WoLF agents used $\frac{\delta_l}{\delta_w} = 2$. Results were averaged over 30 runs, 1 million episodes each. In (a), both agents used the same learning algorithms. For both algorithms, only one player's policy is shown. In (b), one agent used WoLF-PHC and the other used PHC. Both player's policies are shown.

at the convergence of the algorithm in a non-self-play environment.

The policy hill-climbing algorithm (PHC), that unlike JALs or Nash-Q does not keep beliefs on the actions of other agents, extends general Q-learning (Watkins and Dayan, 1992) by keeping track of the current mixed policy. This means that, aside from storing Q-values, PHC also stores π-values that represent the current probability of playing each action. At each timestep a PHC agent selects an action a in state s with probability $\pi(s, a)$ and some rate of exploration ϵ. After receiving reward r for action a the agent updates the corresponding Q-value using the standard Q-learning formula:

$$Q(s, a) \leftarrow (1 - \alpha)Q(s, a) + \alpha(r + \gamma max_{a'} Q(s', a')) \tag{1}$$

Not only does the agent update the Q-value of the action it chose, it also updates all its π-values as follows:

$$\pi(s, a) \leftarrow \pi(s, a) + \begin{cases} \delta & \text{if } a = argmax_{a'} Q(s, a') \\ \frac{\delta}{|A_i|-1} & \text{otherwise} \end{cases} \tag{2}$$

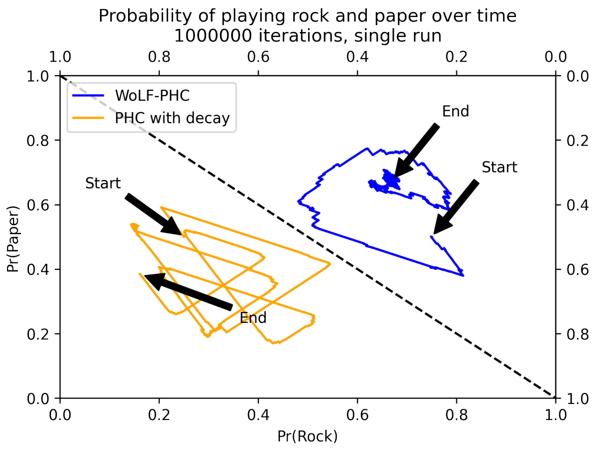
Watkins and Dayan (1992) already proved that Q-learning is rational and that it converges to an optimal policy when all other agents play stationary strategies. Bowling and Veloso state that, since Q-learning is rational, the PHC algorithm is rational as well. However, the Q-learning proof shows that the Q-values converge to Q^* when a reasonable exploration strategy is used. Therefore, the π-values converge to a policy that is greedy according to these Q-values, that converge to Q^*, which are the optimal Q-values. This means that, the PHC algorithm might be convergent when used in an environment of agents that are all rational and play stationary strategies, but there is no proof that it is convergent when the other agents do not follow this description. In section 3 this is proven empirically. The WoLF policy hill-climbing algorithm that Bowling and Veloso introduce tries to add the convergence property to PHC. It extends the PHC algorithm with a variable learning rate that follows the WoLF principle. Implementing such a learning rate allows other players more time to adapt to a change in a player's strategy when it at first seems beneficial. On the other hand, when it is harmful, players are allowed to react more quickly. In practice, to transform the PHC algorithm that was described so far into the WoLF-PHC algorithm, the only thing that needs to change is that now there should be two learning rates δ_l and δ_w that represent respectively the learning rate when losing and the learning rate when winning. Additionally, when the algorithm now updates its π-values using (2), the used learning rate is determined as follows:

$$\delta = \begin{cases} \delta_w & \text{if } \sum_a \pi(s, a)Q(s, a) > \sum_a \bar{\pi}(s, a)Q(s, a) \\ \delta_l & \text{otherwise} \end{cases} \tag{3}$$

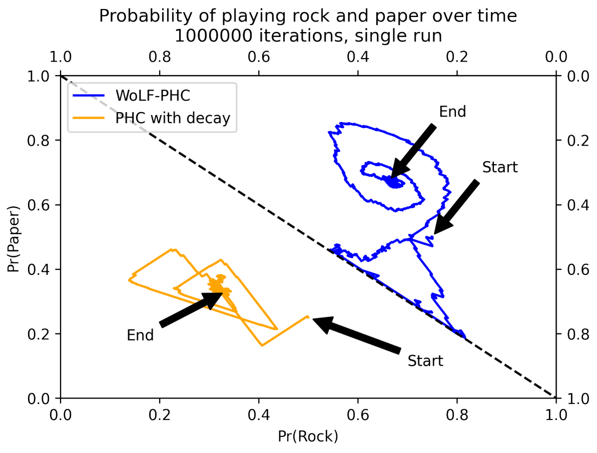
In which $\bar{\pi}$(s, a) is the average policy for playing action a in state s.

3 Results

The four games that Bowling and Veloso used to analyse and compare the PHC and WoLF-PHC algorithms were all two-player games. The first two are simple zero-sum matrix games, which are used to show how the WoLF-PHC algo-



(a) Self-play



(b) One WoLF-PHC agent, one PHC agent

Figure 2: Policy trajectories for the rock-paper-scissors game. For this game, the WoLF-PHC agents used $\frac{\delta_l}{\delta_w} = 2$. These are the results for a single trial of 1 million episodes, for thirty such trials the results were always similar. In (a), both agents used the same learning algorithms. For both algorithms, only one player's policy trajectory is shown. In (b), one agent used WoLF-PHC and the other used PHC. Both player's policy trajectories are shown.

$$\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}$$

(a) Matching pennies

$$\begin{bmatrix} 0 & -1 & 1 \\ 1 & 0 & -1 \\ -1 & 1 & 0 \end{bmatrix}$$

(b) Rock-paper-scissors

Figure 3: Payoff matrices for one agent in the matrix games. The other agent receives the negated reward.

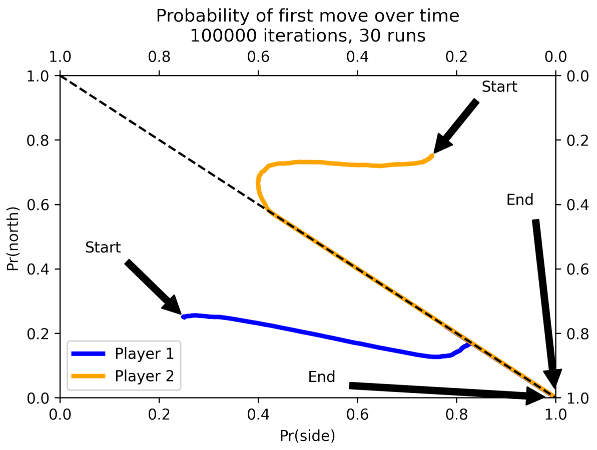
rithm works and what its influence is on convergence. The payoff matrices for these games are shown in Figure 3. For both of these zero-sum matrix games, the equilibrium is a mixed one that consists of playing each action with equal probability. The two other games are more complex multi-state stochastic games, they are explained here briefly.

Gridworld The gridworld game was first introduced by Hu and Wellman (1998). In an instance of the gridworld game, agents start in the corners of a grid and try to reach a specific goal square in the grid. At each time step, an agent can choose between four actions corresponding to the four directions they can move. When two agents try to move to the same square their move fails and the turn is wasted. In most implementations of the gridworld game, the "north" action is made uncertain for agents that are in their initial squares, only executing it with a 0.5 probability. Doing this makes the game more interesting, as going north can now be less optimal than going sideways, thereby forcing players to interact. There are two Nash equilibria in the gridworld game. Both consist of one player moving sideways while the other player tries to move north. The equilibria show that teamwork between the agents can be crucial.

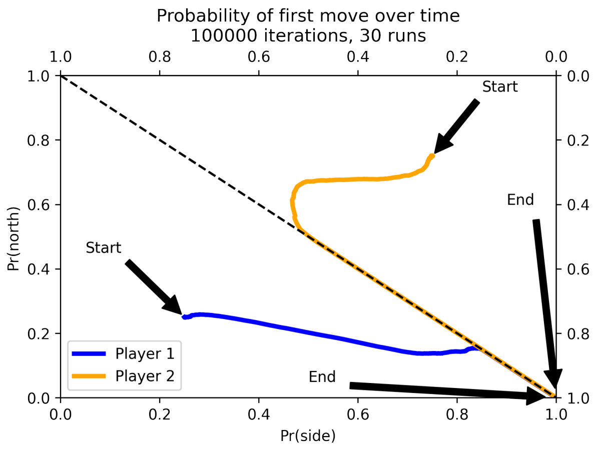
Soccer The soccer game was first introduced by Littman (1994b). In a way it is similar to the gridworld game but there are some important differences. In an instance of the soccer game, agents start at a *random* position in a grid and one of the agents gets assigned the ball. The objective of the game, just as in real soccer, is to bring the ball to the goal of the other agent. Again, at each time step an agent can choose between 4 actions that correspond to the directions it can move and an additional action that corresponds to not moving at all. Note that for this game, the agents choose their actions simultaneously, but the order in which they are executed is random. This way, when an agent chooses an action that would bring them to a square that is already occupied, if they possess the ball, the other player receives the ball and the current player loses their turn. For the soccer game, there is no pure Nash equilibrium only a mixed one.

Simulations

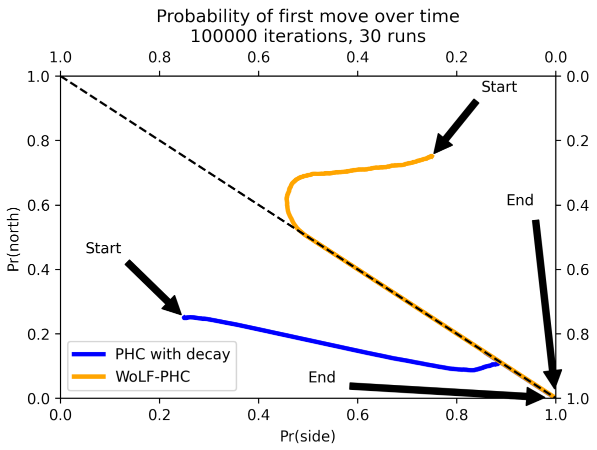
In the simulations the original self-play environments are always reproduced and each time a new experiment for the non-self-play environments is introduced. In the non-self-play environments the decision was made to always let one agent use the WoLF-PHC algorithm while the other agent uses normal PHC. Exploration (ϵ) and discount (γ) rates were chosen such that the results resemble those of the original experiments, the exact values can be found in Appendix A. In all simulations, both α and δ were always decreased over time. Note that the actual decay formulas were not explicitly mentioned in (Bowling and Veloso, 2001) but they were listed in a newer, more complete, transcript (Bowling and Veloso, 2002). The formulas were copied but changed with respect to the values for ϵ and γ,



(a) Self-play: PHC



(b) Self-play: WoLF-PHC



(c) One WoLF-PHC agent, one PHC agent.

Figure 4: Average policy trajectories for both players in the gridworld game. For this game, the WoLF-PHC agents used $\frac{\delta_l}{\delta_w} = 4$. Results were averaged over 30 runs, 10^5 episodes each. In (a) and (b), both agents used the same learning algorithms, PHC and WoLF-PHC respectively. In (c), one agent used WoLF-PHC and the other used PHC.

again to achieve results that resemble those of the original experiments. The exact formulas are provided in Appendix B. In the matrix game simulations, the WoLF-PHC agent always used $\frac{\delta_l}{\delta_w} = 2$. For the more complex games, Bowling and Veloso gave the WoLF-PHC agents a more "aggresive" $\frac{\delta_l}{\delta_w} = 4$. Note that, just as explained in section 2, this means that when an agent is "losing", its learning rate is 4 times bigger than its learning rate when it is "winning". The soccer game experiments were not explicitly executed. An explanation is given on how these experiments would be structured, as well as a speculation on what the results could be, but the focus lies on the gridworld game. This game should be sufficiently complex to allow drawing conclusions.

Matrix games For both matrix games, the results were averaged over 30 trials with 1 million episodes each. Since both matrix games are zero-sum games, the initial policy of the agents needs to be fixed as otherwise averaging would not be representative.

For the matching pennies game Figure 1a shows that, just as in the original experiments, the WoLF-PHC agent quickly starts oscillating around the mixed Nash equilibrium and over time it comes closer and closer. The PHC agent also oscillates but does not show a decrease in amplitude. For this game, in a self-play environment, the WoLF-PHC algorithm seems to be convergent. Interestingly, Figure 1b shows that when the game is simulated with one WoLF-PHC agent and one PHC agent, WoLF-PHC is "smart" enough to guide the PHC agent to the equilibrium.

The results for the rock-paper-scissors game follow the same pattern. Figure 2a shows that the PHC algorithm circles the mixed Nash equilbria without any sign of ever getting closer to it. On the other hand, the WoLF-PHC algorithm spirals to the equilibrium, proving that for this game as well, the WoLF-PHC algorithm is convergent in a self-play environment. When the environment consists of one WoLF-PHC agent and one PHC agent, the more simple PHC agent is again able to converge to the equilibrium. Figure 2b shows that in this case, the WoLF-PHC algorithm, at some point in time, completely stops playing *scissors*. However, due to the exploration rate the action is rediscovered. Adding a third action seems to have made it more difficult for WoLF-PHC to guide the PHC-agent to the equilibrium.

Gridworld For the gridworld game, results were averaged over 30 trials, 100000 episodes each. The exact setup used for the gridworld experiments is given in Appendix C. Figure 4 shows that in a self-play environment, PHC as well as WoLF-PHC are able to converge to a Nash equilibrium. One of the agents learns to go sideways from its initial state while the other agent learns to go north. In the non-self-play environment, even though the agents do not use the same learning strategy, they are still able to converge. These results show that the WoLF-PHC algorithm allows for agents to learn an equilibrium even when applied to complex general-sum games that have multiple equilibria. On top of that, when combined with a normal PHC agent, both still converge.

Soccer As mentioned before, the soccer game experiments were not explicitly executed. However, in what follows, an explanation is given on how the experiments could be structured and their results are speculated.

The soccer game setup from the original experiments was inspired by the minimax-Q experiments from Littman. The details are given in Appendix D. The minimax-Q agents are omitted as for this article, the comparison is irrelevant. For the current analysis, six different cases of training are required:

- Two WoLF-PHC agents.
- Two WoLF-PHC agents with $\delta = \delta_l$.
- Two WoLF-PHC agents with $\delta = \delta_w$.
- One WoLF-PHC agent and one PHC-agent with $\delta = \delta_l$.
- One WoLF-PHC agent and one PHC-agent with $\delta = \delta_w$.

After training the agents for 1 million steps, their policies are fixed and each time one of the agents is chosen – in the case of the non-self-play environments the PHC-agent is chosen – to train against a simple Q-learner. Litmann reasons that doing this determines the chosen agent's policy's worst-case performance and gives an idea of how close the agent got to the equilibrium. In general the equilibrium of the soccer game wins at least half of its games. Bowling and Veloso rightfully state that unlike minimax-Q, WoLF-PHC and PHC generally oscillate around the solution. Therefore, they choose to, after 1 million training steps, train for 250000 episodes more, saving the policies at each 50000 steps. From the five saved policies, the worst policy is then chosen as the resulting policy for that training run. Averaging over 30 training runs should be sufficient. Once the agent's learned policy is known, a simple Q-learner is trained against the fixed policy for another 1 million steps. Lastly, when both agents have learned their policies, they are matched against each other for 50000 matches. The results of the matches can then be plotted using a win-percentage bar plot.

4 Discussion

In this article, most of the results from the original WoLF-PHC experiments on four 2-player stochastic games are reproduced. Thereby reconfirming Bowling and Veloso's findings that the WoLF-PHC algorithm is rational and also, empirically proven for 2-player games, convergent. Additionally, the algorithm's performance is evaluated in non-self-play environments for the same 2-player games. The second player is always implemented using the normal PHC algorithm. The results show that WoLF-PHC is competent enough to converge itself and allow the rational, but not necessarily convergent, PHC algorithm to also converge.

Appendices

Explanations of the algorithms are given in section 2. The appendices give in depth details of the experimental setups. Additionally, the source code can be found on the github repository[1]. All results are thus reproducible. Upon request, the results used for the figures can be provided in the form of *Python pickle files*.

A Non-decaying parameters

The exploration and discount rates were the same for all the experiments.

- $\epsilon = 0.05$
- $\gamma = 0.8$

B Decaying parameters

In what follows, the decaying learning rates are listed, just as they were given in (Bowling and Veloso, 2002), but transformed with respect to the values of ϵ and γ in Appendix A. In the formulas, t always corresponds to the current episode number.

- Matching pennies:

$$\alpha(t) = \frac{1}{100 + \frac{t}{10000}}, \delta = \delta_w(t) = \frac{1}{20000 + t}, \delta_l(t) = 2\delta_w(t)$$

[1] `https://github.com/wulfdewolf/CGTproject`

- Rock-paper-scissors:

$$\alpha(t) = \frac{1}{10 + \frac{t}{10000}}, \delta = \delta_w(t) = \frac{1}{2000 + t}, \delta_l(t) = 2\delta_w(t)$$

- Gridworld:

$$\alpha(t) = \frac{1}{10 + \frac{t}{500}}, \delta = \delta_w(t) = \frac{1}{1000 + \frac{t}{10}}, \delta_l(t) = 4\delta_w(t)$$

C Gridworld game setup

For the gridworld game experiments, the setup was mostly copied from the original experiments. Figure 5 depicts the original grid. Reaching the goal state yields a reward of 100. Every other action results in a reward of 0. When agents are in their initial states, the *north* action is uncertain. With probability 0.5 it results in them actually going *north*, the other 0.5 probability results in them doing a random other action. When one of the agents reaches the goal state, the game resets and both agents are transported back to their initial states.

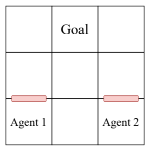


Figure 5: The grid used for the gridworld experiments. The red blocks depict walls for which the corresponding actions are uncertain.

D Soccer game setup

In their experiments, Bowling and Veloso used the soccer grid shown in Figure 6. Litmann originally devised this grid, and the soccer game in general, to evaluate the performance of the minimax-Q algorithm.

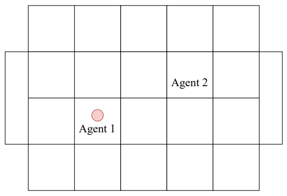


Figure 6: The original soccer field grid.

References

Bowling, M. and Veloso, M. (2002). Computational game theory group project. *Artificial Intelligence*, 136:215–250.

Bowling, M. H. and Veloso, M. M. (2001). Rational and convergent learning in stochastic games. In Nebel, B., editor, *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence, IJCAI 2001, Seattle, Washington, USA, August 4-10, 2001*, pages 1021–1026. Morgan Kaufmann.

Claus, C. and Boutilier, C. (1998). The dynamics of reinforcement learning in cooperative multiagent systems.

Conitzer, V. and Sandholm, T. (2003). AWESOME: A general multiagent learning algorithm that converges in self-play and learns a best response against stationary opponents. *CoRR*, cs.GT/0307002.

Hart, S. and Mas-Colell, A. (2000). A simple adaptive procedure leading to correlated equilibrium. *Econometrica*, 68(5):1127–1150.

Hu, J. and Wellman, M. P. (1998). Multiagent reinforcement learning: Theoretical framework and an algorithm. In *Proceedings of the Fifteenth International Conference on Machine Learning*, ICML '98, page 242–250, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Hu, J. and Wellman, M. P. (2003). Nash q-learning for general-sum stochastic games. *J. Mach. Learn. Res.*, 4(null):1039–1069.

Littman, M. L. (1994a). Markov games as a framework for multi-agent reinforcement learning. In *Proceedings of the Eleventh International Conference on International Conference on Machine Learning*, ICML'94, page 157–163, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Littman, M. L. (1994b). Markov games as a framework for multi-agent reinforcement learning. *Machine Learning Proceedings 1994*, page 157–163.

Shapley, L. S. (1953). Stochastic games. *Proceedings of the National Academy of Sciences*, 39(10):1095–1100.

Verdoodt, A. F., Bontinck, L., Ajridi, S., and De Wulf, W. (2020). Computational game theory group project. GitHub commit: ad25f14b9c98994c19dd2e9bff82834b096b2b4d.

Watkins, C. J. C. H. and Dayan, P. (1992). Q-learning. *Machine Learning*, 8(3):279–292.