UiT The Arctic University of Norway

Faculty of Engineering Science and Technology
Department of Computer Science and Computational Engineering
UiT - The Arctic University of Norway

Learning social codes for self-interested agents

author Gisle Jaran Granmo

Thesis for Master of Science in Technology/Sivilingeniør SHO6264 May 2021

Contents

Abstract

This thesis will explore interaction between non-supervised machine learning agents, with a focus on traffic situations. We intend to show that self-interested agents can arrive at mutually beneficial strategies according to general game theory, achieving pure or mixed Nash equilibria, and that this has practical applications for unmanned vehicles.

We will start by verifying against simple one stage models, specifically "battle of the sexes" and "chicken race", before extending this to uncontrolled four way intersections.

1 Introduction

The field of artifical intelligence and reinforcement learning sees broad applications in a variety of fields; finance, robotics, as well as more leisurely ones like computer games, and chess, and many more. The field has seen many advancements in recent years and getting a full overview of the current state of art is no small task.

The question posed for this thesis is whether self interested agents can achieve mutually beneficial social behaviour. This is studied through applying selected reinforcement algorithms on classic game theory problems, like Prisoner's Dilemma. Being able to achieve mutually beneficial behaviour in limited problems like these is a prerequisite for applying them to more complicated problems. Central to this is observing if the agents can achieve Nash Equilibria and/or converge to pareto efficient policies. Various approaches and algorithms have been used before but there will always be challenges and limitations that are unique to each of them.

After applying Q-Learning and an actor-critic implementation in the two player settings common in game theory, I was able to confirm that some beneficial behaviour was reproducable even with purely self interested behaviour. However in other cases this would lead to one agent taking advantage of the other. As such one must take care when applying these and observe that one actually gets the result that is desired.

2 Game Theory

Game theory is the study of strategical interaction between players, where the outcome for each player is dependant on the actions of all participating players. By using an appropriate model for a given situation, game theory can be used to deduct optimal behaviour for each player. The field has a wide range of applicability, and is commonly used in economics and other social sciences, as well as artifical intelligence.

2.1 Definitions

Common terminology for game theory. [Bošanskýa et al., 2016]

Definition 1. *(Strict dominance)*
A strategy or action is strictly dominated if all other choices lead to a better outcome. Rational players will never chose a strictly dominated strategy.

Definition 2. *(Pure vs Mixed Strategy)*
A strategy may be either pure or mixed. In a pure strategy the same choice is always taken. In a mixed strategy each action is assigned a probability, and selected randomly from this probability distribution.

Definition 3. *(Best response)*
Given that one player follows a set strategy, the best response by another player is the strategy which gives the highest reward.

Definition 4. *(Nash Equilibrium)*
A Nash equilibrium is achieved when no player can improve their outcome by changing strategy, in a non-cooperative setting. That is, each player will make decisions to maximize their own payoff, without regarding the payoff other players receives. Players do however take into account what choices are optimal for other players, and formulate their own best response based on this. A Nash equilibrium may be pure or a mixed strategy.

Definition 5. *(Pareto Efficiency)*
An outcome is said to be pareto efficient if there exists no other outcome that is better for at least one player, without being worse for any other players. An outcome is pareto inefficient if there exists any outcome that is better for at least one player, without being worse for any other players.

A Nash equilibrium is not necessarily pareto efficient, and vice versa.

Definition 6. *(Normal-form game)*
In a normal-form game players act simultaneously without knowledge of the choice of other players.

2.2 Matrix games

Games between two players are commonly described using an action-payoff matrix, where each player receives a payoff depending on the actions of all players. The payoffs are unique to each player and not necessarily comparable between players. Players make their choices without knowledge of the choices of other players but may have knowledge of past events.

Some examples of common normal-form matrix games follows.

2.2.1 Battle of the Sexes

Battle of the Sexes is a two player problem, where the participants wants to agree on the same choice action or strategy, but also have opposing preferences:

A couple wishes to meet for a date; either at the cinema, or at the ballet. One prefers the cinema, the other the ballet. While each have their own preference, both would rather meet than go alone.

Table 1: Battle of the Sexes

	Cinema	Ballet
Cinema	2/1	0/0
Ballet	0/0	1/2

This gives us two pure Nash equilibria, both which are pareto efficient, and one mixed strategy.

2.2.2 Chicken Race

A chicken race is a two player problem where both players lose if neither concedes, and winning is reliant on your opponent conceding out of fear of losing . If both players concede neither receives reward or penalty, though the lack of reward may be considered a penalty depending on which situation the game is meant to illustrate.

Typically this game is portrayed as two cars approaching from opposite directions at speed, daring eachother to be the last to swerve away from the path of collision. The game is only winnable with different choices by the players, otherwise ending either in collision or a draw. Stay/Swerve are Nash Equilibra, and any other choice is pareto efficient.

This again gives us two pure Nash equilibria, and one mixed strategy - however the mixed strategy will always entail the risk of both players chosing to stay on course and colliding.

Table 2: Chicken Race

	Stay	Swerve
Stay	-10/-10	1/-1
Swerve	-1/1	0/0

2.2.3 Prisoner's Dilemma

Two people are arrested and questioned separately. They both are guilty, but it can not proven it without a confession from either prisoner. If both prisoners cooperate and keep quiet they can both only be given a short sentence, so they are given a proposition: If either of them help prove the other prisoner is guilty, they can go free, and the other prisoner will receive a long sentence, but if they both confess they will each receive a medium length sentence.

Table 3: Prisoner's Dilemma

	Quiet	Confess
Quiet	-1/-1	-10/0
Confess	0/-10	-7/-7

In this situation the Nash equilibrium is for both players to confess. Both players can improve their payoff by confessing if the previous choice was anything but confess/confess, and neither can can improve their payoff by switching to keeping quiet from confess/confess. However any other combination of choices are pareto efficient, but not confess/confess.

2.2.4 Matching Pennies

Two players reveals a coin with either heads or tails showing. Player one wins if the coins matches, and player two wins if they differ.

Table 4: Matching pennies

	Heads	Tails
Heads	1/-1	-1/1
Tails	-1/1	1/-1

This game illustrates mixed strategy Nash equilibra. Any pure strategy will always lose, and the only way to maximize payout is to randomly select heads or tails. An alternative game to illustrate this is rock, paper, scissors.

3 Reinforcement Learning

Reinforcement learning mimics natural learning by letting an agent explore its environment and formulate a strategy based on the rewards it receives for performing available actions. The behaviour of an agent is defined by the choice between selecting known best rewards, or exploring new options. A greedy agent will tend to focus in the direction of the immediate best reward it can find, while an agent leaning towards exploration will test a wider array of options. Finding the correct balance is cruicial for optimimal performance. The formal solution for this is the Markov Decision Process (MDP). [Yang and Wang, 2021]

Definition 7. *(Markov Decision Process)*
A Markov Decision Process is defined by the tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$

- $\mathcal{S}$*: State space. The states that are available to an agent.*
- $\mathcal{A}$*: Action space. The actions that are available to an agent.*
- $\mathcal{P} : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ *is the transition probability from state s to any state s' for a given action.*
- $\mathcal{R}$*: Reward function. The payoff an agent receives for a transition from (s,a) to s'.*
- $\gamma \in [0, 1 >$*Discount factor.*

[Yang and Wang, 2021]

3.1 Model based vs model free

A reinforcement learning algorithm is either model based or model free. A model based algoritm is aware of the states and transitions that are available, while a model free algoritm learns about its environment through exploration.

3.2 Value based vs policy based

Algortithms can be divided into two categories, value based and policy based, with some overlap. A value based algorithm seeks to maximize the value (Q) from its available actions in a given state. A policy based algorithm tries to find an optimal probability distribution of the available actions in that state.

4 Multi Agent RL Systems

In a multi agent setting an agent can not only observe its environment but must also take into account the actions performed by other agents. This adds both uncertainty and complexity. S [Yang and Wang, 2021][Zhang et al., 2021]

4.1 Markov Games

Definition 8. *(Markov Game or Stochastic Game)*
A Markov Game is defined by the tuple $\langle \mathcal{N}, \mathcal{S}, \mathcal{A}^i, \mathcal{P}, \mathcal{R}^i, \gamma \rangle$

- $\mathcal{N}$*: Number of agents. N=1 is a single-agent Markov Decision Process as describer earlier.*
- $\mathcal{S}$*: State space. The states that are available to all agents.*
- $\mathcal{A}^i$*: Action space for agent i. The actions that are available to that agent.*
- $\mathcal{P} : \mathcal{S} \times \mathcal{A} \rightarrow \Delta(\mathcal{S})$ *is the transition probability from state s to any state s' for a given combination of actions by all agents.*
- $\mathcal{R}^i$*: Reward function. The payoffs each agent receives after a transition from (s,A) to s'.*
- $\gamma \in [0, 1 >$*Discount factor.*

[Yang and Wang, 2021]

4.2 Scalability

One significant obstacle in tackling MARL is the exponential nature of the action space. Even for a simple game with two actions per player, we will have a doubling of the action space for each player added. Larger numbers of actions per player will naturally only compound this issue. Even solving two player games for Nash Equilibria can be non trivial. An important field of study is limiting the scope that is necessary for each agent to perform, so that we can deal with a subset of the action space, rather than the entire action space. [Yang and Wang, 2021]

4.3 Uncertainty

When multiple agents operate in or otherwise affect the same state space it becomes non-stationary. If the agents are also learning as they interact with the environment, their behaviour will also most likely change over time, both with and without interaction with other agents. Therefore a choice must be made if and how we deal with this. Options include but are

not limited to simply treating them as stationary, trying to learn the opponents, or even assuming the opponents are trying to learn your behaviour. [Hernandez-Leal et al., 2019]

4.4 Approaches

For practical part of this paper two of the simpler algorithms have been explored. There are many more notable algorithms that are possible to approaches MARL problems with. Of note would be Deep-Q Network, Nash-Q Learning, and building on them Nash-DQN.

5 Materials & Methods

5.1 Software used

For the simulations TensorFlow and Keras were employed. TensorFlow provides functionality for performing necessary computation both on CPU and GPU in various configurations. Keras provides a convenient Python API wrapper for TensorFlow.

For ease of installation premade Docker images were used. The image used (latest - TensorFlow 2.4.1 at the time of download) included Jupyter Notebook/Python, and various necessary Python libraries like Numpy, and as such provided a complete development setup in a single image. All code from the project is supplied as Jupyter notebooks with Python as development language, and can be run on the previously mentioned Docker image. Docker images for TensorFlow are available at https://www.tensorflow.org. For the Keras API see https://keras.io/.

Keywords

- TensorFlow
- Keras
- Docker
- Python
- Numpy

5.2 Software considered

At the start of the project visual simulation of traffic in SUMO/Flow/RLlib was considered as an end goal. But due to time constraints this was omitted, and focus was shifted to TensorFlow, which is one of the libraries that this software stack is built upon. During the development it became readily apparent that for this project OpenAI Gym would have been a useful addition to the software selection. Many of the official examples for TensorFow/Keras utilised the premade environments from OpenAI Gym. To best utilize them simplified environments functionally similar to OpenAI's Gyms were writen.

SUMO/Flow stack is availabe from https://flow-project.github.io/, and OpenAI Gym from https://gym.openai.com/

Keywords

- SUMO

- Flow
- RLlib
- OpenAI Gym

5.3 Testing and simulation

The selected reinforcement algorithms were first applied to the same set of 2x2 matrix games to test applicability. Number of episodes and episode lengths were chosen pragmatically by observing which number of sample sizes were sufficient to produce stable behaviour for each algorithm. Average reward per episode was considered the most important metric as it easily showed if the agents were able to achieve similiar rewards from the same game. For Q-learning the actions taken on the last episode were used to evaluate the policy the agents had converged to. For Actor-Critic the action probability distribution was used to show policy of the agents. Since this was 2x2 matrix games, showing the probability of one choice was sufficient as the probability of the other would be complimentary.

6 Simple Matrix Games

For initital testing we consider the four matrix games outlined earlier in the report; Battle of the Sexes, Prisoner's Dilemma, Chicken Race, and Matching pennies. These 2x2 games provide a convenient testing environment for core functionality of learning algorithms, and can show the ability or inability of an algorithm implementation to reach pure and/or mixed Nash equilibra.

6.1 Two-agent Joint-Action Q-learning

Q-learning was chosen as a starting point due to it's simplicity. It is a value based algoritm, and as such does not provide stochastic policies by default, but is still able to find satisfactory solutions for some of these games. Previous actions taken by both Agents were observable as part of the state (Joint-Action), enabling the agents to adapt to their opponents strategy.

For Q-learning in 2x2 games patters were observable after a relative small number of iterations. For each matrix game 20 episodes of 100 repeated games were ran, with ϵ set to 0.9 for the first 25

6.1.1 Battle of the Sexes

For the Battle of the Sexes game, the agents arrive at one of the pure Nash-Equilibra, but without deviation to the other. The result being that one agent consistently outscores the other.

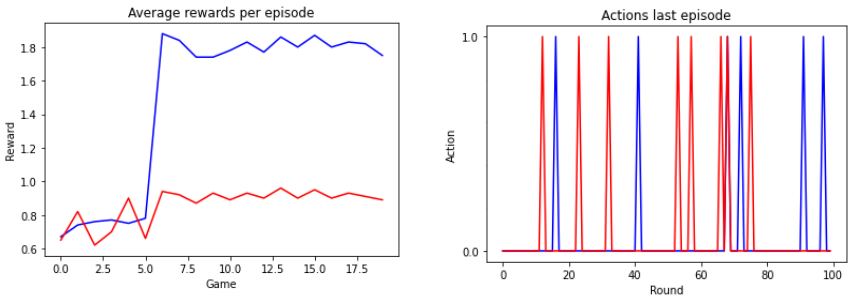

Figure 1: Battle of the Sexes, ϵ-greedy Q-learning

6.1.2 Prisoner's Dilemma

In the Prisoner's Dilemma game the agents both adopt a mutually beneficial strategy, cooperating by default, and being forgiving when the the inevitable random defection occurs.

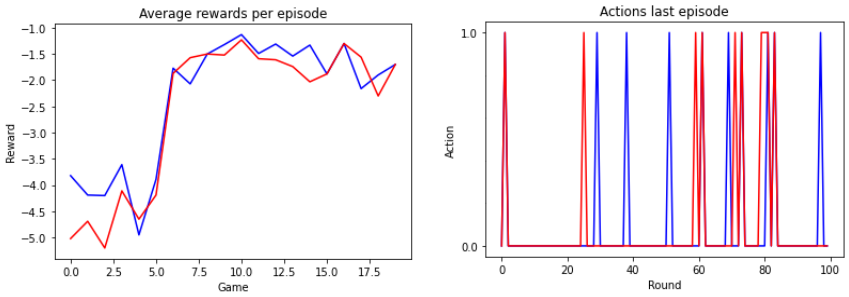

Figure 2: Prisoner's Dilemma, ϵ-greedy Q-learning

6.1.3 Chicken Race

For the Chicken Race game the agents also reach a mutually beneficial strategy, with both yielding. If this is desirable behaviour is debatable. Clearly avoiding a collision is good, but neither agent will ever win.

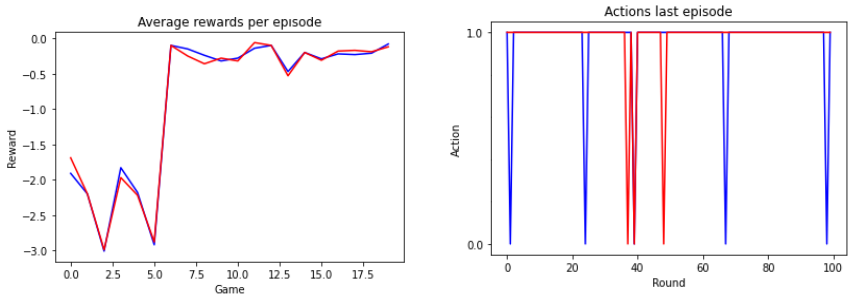

Figure 3: Chicken Race, ϵ-greedy Q-learning

6.1.4 Matching Pennies

As expected the inability of a greedy policy to adopt a mixed strategy clearly shows for Matching Pennies. The agents performed more mutually beneficial during the first 25

6.2 Actor-Critic

For the second algorithm the actor-critic [Bhatnagar et al., 2009] approach was chosen. Actor-critic algorithms are gradient policy based, and as such can provide solutions to problem that have no pure Nash equilibra (basic examples of this being Matchin Pennies or Rock-Paper-Scissors). Actor-critic algorithms generally take longer to converge than Q-learning algorithms, and for the tests 30 episodes of 100 steps were run for each game.

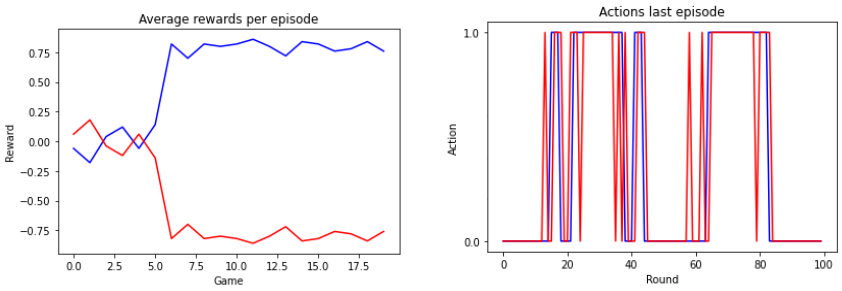

Figure 4: Matching Pennies, ϵ-greedy Q-learning

The implementation was modified from a single agent problem provided on the offical Keras website, https://keras.io/, but did not perform quite as expected, though it was mostly able to reach reasonable policies. However, it did not produce the expected mixed strategy for the Matching Pennies game, which suggests that the specific actor-critic algorithm was not suitable for this kind of game, or inadequately implemented by me.

From the simulation of our matrix games average rewards per episode is plotted per player, and the probability of chosing action $a_0 \in [a_0, a_1]$ at the current state as a representation of the policy.

6.2.1 Battle of the Sexes

We eventually see the same convergence towards a single pure Nash equilibra similar to Q-learning. Notably though, the agents struggled to leave the non scoring states until after 20 episodes had passed. Eventually the agents settled for once of the uneven payoffs, not reaching mixed strategy that could have offered an even reward over time.

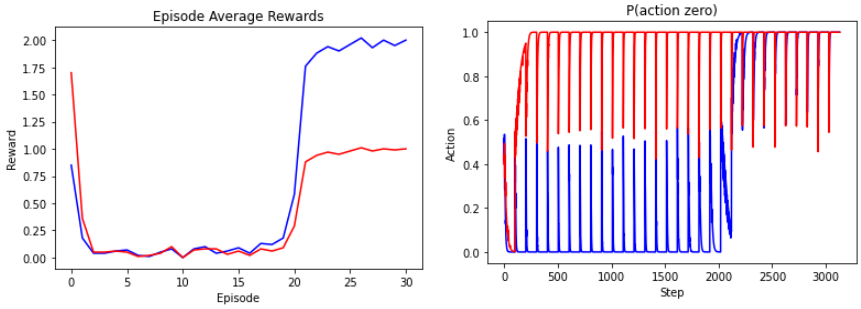

Figure 5: Battle of the Sexes, actor-critic

6.2.2 Prisoner's Dilemma

Here the agents quickly adopt a forgiving cooperation strategy, maximizing both their scores.

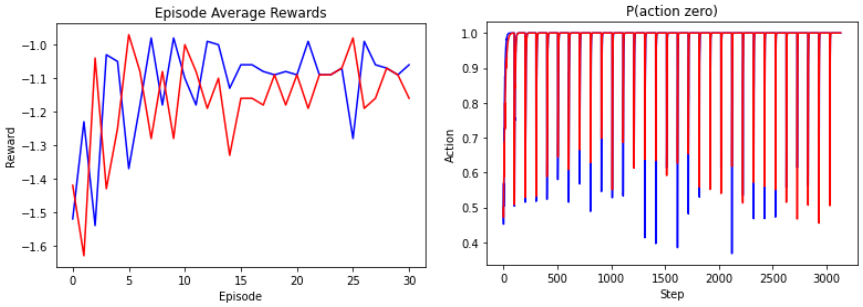

Figure 6: Prisoner's Dilemma, actor-critic

6.2.3 Chicken Race

Chicken Race converges to opposing choices and an uneven score. In contrast to Q-learning we do not converge on the possibly non-resolution choice of yield/yield, but instead one player stays course while the other yields. The fatal stay/stay option is avoided.

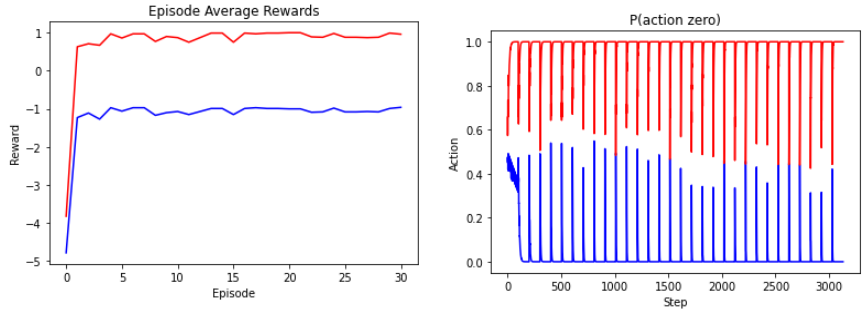

Figure 7: Chicken Race, actor-critic

6.2.4 Matching Pennies

Here a 50/50 policy was expected, but we again see one player dominating the other. The switch between winner and loser isn't reproducable in subsequent simulations, but other similar random anomalies do occur.

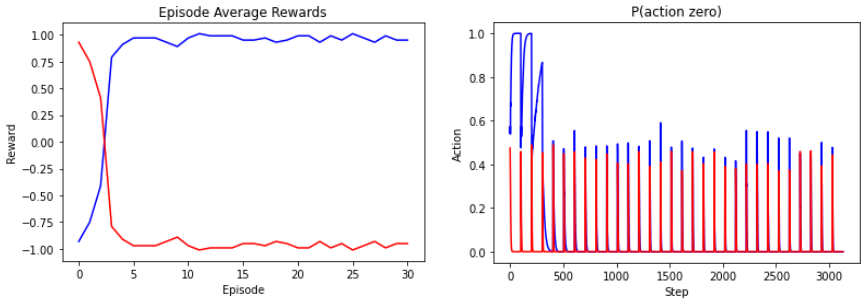

Figure 8: Battle of the Sexes, actor-critic

7 Case studies

7.1 Traffic on crossing roads

Given a simple road cross intersection with no signals or yield signs, crossing traffic may give rise to conflicts of right-of-way.

An easy and safe answer for a single autonomous agent could be to always yield in a situation like this, and let other drivers go first. However, if multiple vehicles are autonomous, and all yield, deadlock situations could arise. The immediate thought might be that communication between the vehicles is required, but can autonomous agents with no communication arrive at a solution simply through observation and experience?

In a real world situation it is unlikely, but not impossible, that several vehicles can arrive at the same time. In most cases the vehicles would reach the intersection with at least a small variation in speed and arrival time. Observing the other vehicles speed and acceleration could help determine policies for safe resolution of the situation, so that the vehicle that expects to arrive last would benefit from yielding for the other vehicles rather than challenging them.

Simplified these cases can be viewed as a finite repeated Chicken Race games. As long as two or more cars choses to both stay their course or yield, the game will be repeated, but there is a limit to the number of games that can be played before reachng the intersection and collision is unavoidable.

7.2 2-way crossing traffic

The 2-way crossing traffic situation can be simplified as a finite repeated 2x2 chicken race. The simulation was run for 50 episodes of 10 steps each. To incentivize the agents to avoid perpetual yielding a penalty to yield/yield was added to the last step. Initial episodes sees attempts of staying course for both agents, until one adopts a yielding strategy to avoid collision. The small

set size results in no agent being able to attain a significantly positive score, though the car adopting the aggressive policy consistently scores highest.

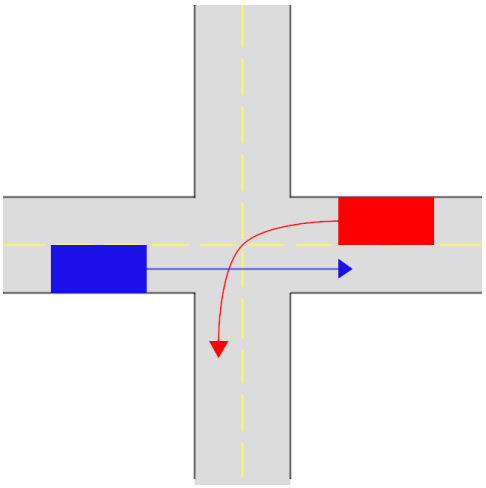

Figure 9: 2-way crossing traffic

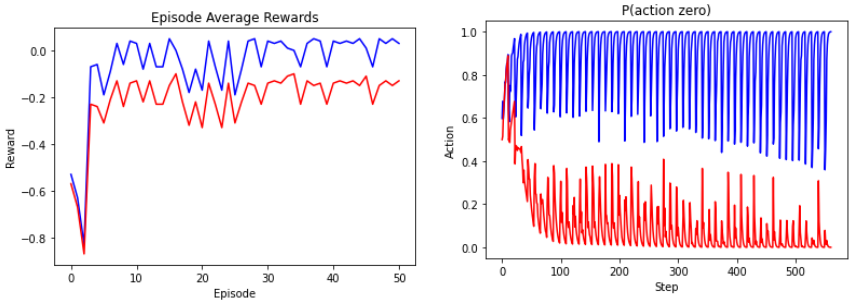

Figure 10: 2-way traffic, actor-critic

7.3 Crossing traffic with n>2 vehicles

For situations with more than two vehicles scalability issues are immediately apparent. Consider the simplified two-action Chicken Race, but add another player as per the three car figure. There is now an action space of 8 combinations. Of these 8 combinations, 4 of them, ie. any with 2 or more cars, may result in a collision, possibly even preventing any yielding vehicle from completing the trip. To successfully negate the conflict a single vehicle must cross each step, and all three vehicles must take into account two other vehicles.

The straight crossing case of four vehicles is similiar to the 3 vehicles with crossing paths, with the exception that there are now 4 vehicles that

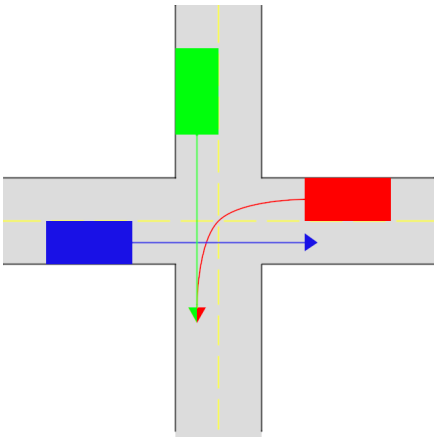

Figure 11: 3-way crossing traffic

must take into account 2 other vehicles. This is thankfully just a linear increase in complexity, but it is of course the simplest case of 4 vehicles in a crossing. A real world scenario with all vehicles having multiple optional paths, and likely many more vehicles in short succession, the issue is only further complicated.

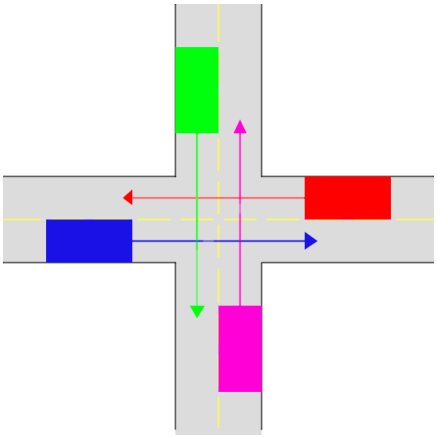

Figure 12: 4-way crossing traffic

8 Conclusion

The field of Multi Agent Reinforcement Learning is large and rapidly expanding in recent years. In this paper I have explored two basic algorithms and approaches to the problem. Challenges involved in scaling algorithms from one to two agents, and from two to more agents, quickly became apparent. Things that seemed trivial at the onset were, of course, not. The array of algorithms and varietes thereof was bewildering at first, and getting a complete overview is a daunting task. The most difficult part was in many ways asking the right questions. In this respect the paper by Yaodong Yang and Jun Wang, "Overview of Multi-agent Reinforcement Learningfrom Game Theoretical Perspective", proved a particularily good source for further study. My only regret in that regard is not coming across thas paper sooner, and I would recommend anyone interested to start their studies there.

Through the implemenation of the two alorithms chosen, Q-learning, and Actor-Critic, I was able to observe mutually beneficial behaviour in some cases. Particularily the case of Prisoner's Dilemma and Chicken Race reached promising results. Of those two Chicken Race was the most applicable to the practical cases I had envisioned, and I would have liked to implemented it in a model more closely based on real life situations.

Time constraints limited me to only implement my algorithms for two agents, but it was clear that scaling up the implementations to more than two participants would not be trivial. In all likelyhood other algorithms would in hindsight look more promising, but one can argue that the insight would not have been gained without them. In particular I would like to explore the use of Nash-Deep Q Networks for the same problems as the simpler algorithms was applied to.

Finally I wish to thank avisor Bernt Bremdal, and Rune Dalmo at the faculty, for assistance and patience with me through my work.

References

[Bhatnagar et al., 2009] Bhatnagar, S., Sutton, R. S., Ghavamzadeh, M., and Lee, M. (2009). Natural actor–critic algorithms.

[Bošanskýa et al., 2016] Bošanskýa, B., Lisý, V., Lanctotb1, M., Čermáka, J., and H.M.Winands, M. (2016). Algorithms for computing strategies in two-player simultaneous move games.

[Hernandez-Leal et al., 2019] Hernandez-Leal, P., Kaisers, M., Baarslag, T., and de Cote, E. M. (2019). A survey of learning in multiagent environments:dealing with non-stationarity.

[Yang and Wang, 2021] Yang, Y. and Wang, J. (2019-2021). An overview of multi-agent reinforcement learning from game theoretical perspective.

[Zhang et al., 2021] Zhang, K., Yang, Z., and Basar, T. (2019-2021). Multi-agent reinforcement learning: A selective overview of theories and algorithms.