University of Southampton

# UNIVERSITY OF SOUTHAMPTON

Faculty of Engineering and Physical Sciences

Electronics and Computer Science

**The Design and Development of Norm-Aware Hanabi Agent**

by

**Pooja Vijayakumar**

**31.07.2023**

**Supervisor: Doctor Vahid Yazdanpanah**

**Second Examiner: Professor Mahesan Niranjan**

A dissertation submitted in partial fulfilment of the degree of MSc Artificial Intelligence

# ABSTRACT

In the case of multi agent systems and multi agent cooperative games, two or more agents involved must learn to cooperate with each other to achieve a common goal and reward. There may also be instances where these agents may need to cooperate with humans to achieve a goal. Under these circumstances, an artificial intelligence (AI) agent may fail to work with human to achieve a task because they do not abide to norms and conventions that humans follow when they interact with other humans. For example, a human would understand that they need to slow down and stop when the traffic light turns yellow when driving on the road and they also assume that other humans abide by this convention when driving on the road. However, this behaviour would not be learnt by an AI agent unless it is trained multiple times to recognize these norms using AI techniques such as reinforcement learning. The purpose of this project is to study how agents that are aware of norms can be built in game setting together with reinforcement learning. Hence, this project will first focus on defining norms in general and then defining normative systems. Then, different types of multiagent systems and multiagent games will be studied to learn different methods of solving game problems. Next, different learning methods will be studied to discover and understand algorithms that can be used to train an AI agent that can imitate human norms and conventions. Finally, an AI game playing agent was built using reinforcement learning and norm aware algorithm that can play the game of Hanabi based on human conventions and norms. The performance of the AI agent was evaluated to study the effect of adding norm aware algorithm on reinforcement learning methods of playing games. Learning to play games while being aware of norms can be extended to real world scenarios such as building self-driving cars. This can be done by modelling an algorithm that can recognize rules and conventions on the road so that the AI agent can learn to drive on the road better using this algorithm while still following human norms and conventions.

# ACKNOWLEDGEMENTS

I would like to acknowledge and express gratitude towards my supervisor, Doctor Vahid Yazdanpanah for expertise and continuous support in completing this research project. His guidance helped me in completing this research project by giving advice on how to solve specific problems and well as resources on research papers related to the topic matter. I would also like to thank my friends and family for their continuous support when developing the necessary algorithms as well as writing the dissertation.

**Statement of Originality**

- I have read and understood the ECS Academic Integrity information and the University's Academic Integrity Guidance for Students.
- I am aware that failure to act in accordance with the Regulations Governing Academic Integrity may lead to the imposition of penalties which, for the most serious cases, may include termination of programme.
- I consent to the University copying and distributing any or all of my work in any form and using third parties (who may be based outside the EU/EEA) to verify whether my work contains plagiarised material, and for quality assurance purposes.

***You must change the statements in the boxes if you do not agree with them.***

We expect you to acknowledge all sources of information (e.g. ideas, algorithms, data) using citations. You must also put quotation marks around any sections of text that you have copied without paraphrasing. If any figures or tables have been taken or modified from another source, you must explain this in the caption and cite the original source.

**I have acknowledged all sources, and identified any content taken from elsewhere.**

If you have used any code (e.g. open-source code), reference designs, or similar resources that have been produced by anyone else, you must list them in the box below. In the report, you must explain what was used and how it relates to the work you have done.

**I have not used any resources produced by anyone else.**

You can consult with module teaching staff/demonstrators, but you should not show anyone else your work (this includes uploading your work to publicly-accessible repositories e.g. Github, unless expressly permitted by the module leader), or help them to do theirs. For individual assignments, we expect you to work on your own. For group assignments, we expect that you work only with your allocated group. You must get permission in writing from the module teaching staff before you seek outside assistance, e.g. a proofreading service, and declare it here.

**I did all the work myself, or with my allocated group, and have not helped anyone else.**

We expect that you have not fabricated, modified or distorted any data, evidence, references, experimental results, or other material used or presented in the report. You must clearly describe your experiments and how the results were obtained, and include all data, source code and/or designs (either in the report, or submitted as a separate file) so that your results could be reproduced.

**The material in the report is genuine, and I have included all my data/code/designs.**

We expect that you have not previously submitted any part of this work for another assessment. You must get permission in writing from the module teaching staff before re-using any of your previously submitted work for this assessment.

**I have not submitted any part of this work for another assessment.**

If your work involved research/studies (including surveys) on human participants, their cells or data, or on animals, you must have been granted ethical approval before the work was carried out, and any experiments must have followed these requirements. You must give details of this in the report, and list the ethical approval reference number(s) in the box below.

**My work did not involve human participants, their cells or data, or animals.**

*ECS Statement of Originality Template, updated August 2018, Alex Weddell aiofficer@ecs.soton.ac.uk*

# Table of Contents

# Table of Tables

# Table of Figures

# Chapter 1 Introduction

## 1.1 Background

Norms are set of rules that an agent abide by and expect others to follow [1]. For example, a human would understand that they need to slow down and stop when the traffic light turns yellow when driving on the road and they also assume that other humans abide by this rule when driving on the road. Norms can also be defined as informal rules that lead the way a society acts [2]. In this case, each agent in a set of agents will form rationale and coordinate with others by considering other agents' beliefs, intentions and desires.

Humans make these inferences intuitively and frequently theorize the behaviour of other humans in a coordination setting from just utilizing sparse observations [3]. This line of reasoning can be extended to concept of Theory of Mind (ToM) which is used to conclude others' actions and thoughts [4]. ToM can also be used to infer how a person acts based on these actions and thoughts of others. Hence, it is important to model how humans' reason about taking actions when cooperating and coordinating with other humans and this can be done by capturing the rules and norms under which an action is taken by people.

Reinforcement learning (RL) is a branch in artificial intelligence (AI) that involves an agent learning a behaviour by interacting with the environment through trial and error while receiving rewards for different behaviours in the environment. However, these agents have no prior knowledge of norms or how to adopt policies that are suboptimal but follows normative behaviour of humans in some settings. In multi-agent reinforcement learning (MARL), agents need to compete to gain a reward or cooperate to achieve a common goal. To achieve common objectives, a set of agents need to coordinate their steps and these agents need to use norms to learn collective actions [1].

This research focused on trying to build agents that learns to cooperate in multi-agent setting using reinforcement learning while being aware of norms. The applications of AI are widely investigated in the domain of games as the environment in games are controlled yet interesting and challenging to be solved [5]. As a result, the focus of this research was to apply norm aware algorithm with reinforcement learning in the cooperative multi-player game of Hanabi. This work can then be extended to real-life scenarios where cooperation with humans and AI are present for example in autonomous vehicles. Norms are prevalent in human society especially when driving on the road and it is important to impose these norms to autonomous vehicles. Autonomous vehicles and human-driven vehicles that can cooperate to improve altruistic social utility can co-exist easily and ensure the well-being on the road [2].

## 1.2 Hanabi

Hanabi is a game played between two to five players. There are a total of 50 cards in deck. If there are two players, each player will get 5 cards and if there are 3 to 5 players, each player will receive 4 cards. All players are allowed to look at the cards of other players but not their own. Each card has a rank and a colour. The possible ranks are 1,2,3,4, 5 and the colours could be red, yellow, green, blue, or white. For each colour, there are three 1's, two 2's, two 3's, two 4's and one 5. There are 3 life tokens and 8 information tokens. A firework is set of cards of same colour with ranks 1 to 5 and there can only be one firework for each colour. The firework must be played in the right order of rank for each colour. The number of points associated with a card is the rank of the card. For example, a rank 2 card is worth 2 points. The goal of the game is to form 5 fireworks, one for each colour. Hence, the maximum number of points possible is 25. The game proceeds in rounds where each player has 3 options,

i. Hint a card.
ii. Discard a card.
iii. Play a card.

Players cannot directly communicate with each other. If hint a card option was chosen, player can provide rank hint or colour hint to another player by pointing at cards that match the ranks or colours. An information token is removed when hint action is done. Hence, hint action can only be done if there are information tokens left. If player choose discard, player can discard a card and get a new card and this in turn cause and information token to be returned. At any time during the game, the maximum number of active information token is 8. If player choose to play a card, then player can do so by placing the card face up and the player gets a new card. The card has been played successfully if it starts a new firework or extends to a current firework. If the card does not fit the firework, then the card is discarded, and one life token is removed. The players can retrieve one information token as bonus every time a firework is completed given that the maximum number of information tokens is 8. The game ends when all 5 fireworks are completed, there are no cards on deck to draw from or if there are no life tokens left. The final score will be 0 if there are no life tokens left or the sum of values in each firework made.

## 1.3 Research aims and objectives

The primary aim of this project was to identify different types of norms present in the game Hanabi and develop a novel algorithm that can learn the norms present in that game. To implement a norm aware algorithm that can imitate human behaviours, a clear definition of how norms and normative systems can be defined needed to be concluded and this was done by studying past research papers on behaviour and actions that would count as norms for humans when they cooperate with each other. Then, the categories

of norms obtained were implemented in the game of Hanabi by training an algorithm that that detected these norms.

The secondary aim of this project was to develop an agent that could play the game of Hanabi using RL in a human-like manner by utilizing the norm aware algorithm. Using reinforcement learning enabled the agent to determine actions that would maximise the rewards it will receive from the environment while the norm aware algorithm helped the agent to play game while being aware of norms in the game and agents' interactions. The developed agent can then be tested on new observations of the game to study the performance of the agent on new instances of data. The performance of the trained agent in the environment helped to determine how well the agent with the aid of norm aware algorithm can play the game of Hanabi by the amount of average reward collected by the agents from playing in the environment.

The objectives of this project were as follows:

- To identify different types of norms, present in human social settings. Past research papers were collected and analysed on norms and normative systems to learn how humans act in cooperation settings. This helped to fulfil the primary aim by understanding what types of norms are present in human society.
- To collect data of agents playing the game Hanabi in an environment based on a set of pre-defined norms learnt during literature review. The data was used to train an algorithm that can detect normative actions based on observations from the environment. This objective aided in fulfilling the primary aim of building a norm aware agent.
- To develop and train an AI agent that played Hanabi based on reinforcement learning while being aware of norms using observations from the environment. This objective helped to fulfil the secondary aim of building a game playing Hanabi agent.
- To evaluate the performance of the AI agent on new observations based on new simulation of the game in the game environment with and without the norm aware algorithm. This helped in fulfilling the secondary aim by determining how well the addition of a norm aware algorithm changed the total rewards received by Hanabi game playing agents.

## 1.4 Related Work

Norm learning is important to enable cooperation with human by building a system that makes interpretable decisions and having an algorithm with policy that is can be adapted to humans' strategies. In [6], the authors argued that there are structure present in strategies of different agents in multi-agent games and they utilised tensor decomposition to find low-rank subspace and find partners' strategies in this subspace. However, this method only focuses on finding policies that imitates behaviour of existing agents and does not extend to learning the normative behaviour that are present in games itself.

Köster et al. [7] is in particular related to this work. The authors introduced norms such as wearing a tie as "silly rules" and build algorithm that can learn these types of norms using reinforcement learning. In particular, they used a foraging environment in which agents needed to avoid poisonous berries. The authors manipulated the environment dynamics by introducing negative rewards for norm violations and they argued that other agents are able learn from these rules enforced upon an agent by observation and learning taboo behaviour. The work in [7] however, mainly focuses on how to enforce norms and build agents that can comply to norms and does not focus on defining different types of actions that would account as norms in general.

In [8], the authors used a technique to introduce normative prior into RL agents so that they could complete tasks while following a normative behaviour. Here, they separated environment reward for completing tasks and normative rewards which is based on how normative an action is. They labelled the normative reward as intrinsic behavioural signal and the environmental reward as extrinsic behaviour signal and used policy shaping to train RL on regular environmental reward while also re-ranking the action choices at every step to encourage agents to take different actions that are normative [8]. In this project, the process of building a norm aware algorithm and RL agent will be separated and the novel contribution in this project was defining how different norms can be defined in terms of actions.

## 1.5 Project structure

In chapter one, the background regarding the research topic was discussed. In chapter two, literature reviews on past research papers regarding norms, normative multi agent systems, multi agent games and learning in game playing AI agents as well as imitation learning were done. This helped to determine how multi-agent systems work and different methods of building game playing agents using RL. This also aided in determining how to model the Hanabi game. Finally, completing literature review helped in narrowing down different types of norms in human society and how a norm aware algorithm can be built. In chapter three, the research methodology of playing game using reinforcement learning while being aware of norms was studied based on the literature review done in chapter 2. Here, the model of the Hanabi game was discussed and 3 different types of norms were defined. The details of building a norm aware algorithm, Hanabi game playing agent and final norm aware Hanabi agent were discussed. Using different methods defined in chapter 3, the implementation details of the norm aware algorithm, the Hanabi game playing agent and the final norm aware Hanabi agent were discussed in chapter 4. Based on the implementations, the results produced were also analysed and the limitations of the implemented methods were discussed. In the following chapter, the problems with implementation of the model were examined based on the implementation details in chapter 4. In chapter 6, the possible future works on improving the performance of the algorithm were discussed.

# Chapter 2 Literature Review

To build a norm aware agent, literature reviews were done to identify types of norms present in human society and are suitable to be incorporated in a norm aware Hanabi game agent. Then, different research papers on multi-agent systems and normative systems were studied to assess how multi-agent systems behave. This fulfilled the primary aim and first objective of identifying norms and normative systems. The aspect of multi-agent systems helped to determine how norms and normative systems can be extended to multi-agent systems as Hanabi is a multi-player game. Next, literature reviews on different games and game learning techniques were assessed to identify methods of building a game playing Hanabi agent. This aided in fulfilling secondary aim and the third objective. Hence, based on these literature reviews, an agent that can play the game of Hanabi while being aware of norms can be developed.

## 2.1 Norms in human society

The process of building a norm aware agent first involved identifying what those norms are in human society and how humans make decision in their everyday life. Then, relevant norms were selected to be built in the norm aware algorithm. Norms are important in social life of humans when interacting with others. Horne and Mollborn [9] categorized norms as group-level judgements of conducts. In terms of judgements, there could be positive connotations or negative connotations associated to norms. In terms of group-level aspect of norms, this can be seen as other individuals accepting or opposing a behaviour, warranting it and seeing such warrants as admissible [9]. Norms can encourage or discourage an individual's way of acting in a society based on feedback the individuals receive from the corresponding actions.

There are many different types of norms present in a human society. According to [10], norms can be categorized as moral norms, rule norms, social norms, and prudential norms. Moral norms depend on internal values of an individual and it is dependent on an individual's moral compass. Rule norms are imposed by officials or establishment based on common understanding between individuals involved such as paying taxes [10]. On the other hand, prudence norms are involved when individuals make decision based on common sense [10]. Finally, social norms are actions that are deemed appropriate within a group of people such as greeting a friend. Learning social norms present in a society involves studying how social interactions are carried out in different social settings.

During decision making, social norms play an important role in determining responses that are acceptable in during action selection process for a group of people [11]. Individuals considers how their action would cause a positive or negative impact on themselves and other individuals when taking certain actions. While it is rational for a person to only consider their own needs when taking actions, this would eventually lead to decrease in common good among a group of individuals which would leave everyone worse off than if everybody cooperated [12]. Hence, people conform to norms as they are motivated to maintain their sense of belonging to their social groups, feel like they have a correct understanding of social facts and steer clear of social rejection [13]. Since Hanabi is a cooperating game where players must work together to achieve maximum reward, social norms are relevant to be built in the norm aware agent. Like humans, here an agents must consider how their actions affects other agents and in future, the overall reward.

The type of social norm that we learn may vary depending on situation [14]. Hence the type of norms in one state an individual is in my not be applicable in another state, for example in different countries. In Hanabi, the type of actions taken may also change depending on the episode the agent is currently in. According to [15], individuals adopt an action that is taken most frequently within a group in a social setting. In this case, individuals adopt a frequency-dependent strategies when selecting an action [15]. This could be extended to the game of Hanabi where an consider the most frequent action taken in a game. The social norm adopted by an individual also depends on the behaviour of the other individual they are interacting with. When socializing with others in new settings in which there is certain normative behaviour to follow, individuals will heed others' response and monitor themselves to behave suitably [14]. Once the social norms have been identified, individuals learn these norms through reinforcement learning by acting on these norms and receiving feedback through rewards [14]. In Hanabi game, positive and negative reinforcement in terms of rewards could help the agent to find optimal game strategy.

A social dilemma is a state in which everyone in a group receives a bigger outcome if each individual chase the common group interest, but every individual in the group is better off than all group members if they all chase the common interest [12]. Thøgersen [16] stated that social norms play a crucial part in fostering cooperation in social dilemmas. As conclusion, social norms play a vital role in instilling cooperation and enabling individuals to work together to achieve common goals and the different type of social norms in human society can be extended to the game of Hanabi. If agents in the Hanabi game can learn to cooperate together based on these social norms, then can receive a larger final reward.

## 2.2 Multi-agent systems and normative systems

Multi-agent systems (MAS) are methods of working out solutions to complex problems by diving them into smaller problems and these smaller problems are allocated to individual agents to be solved [17]. These agents then determine the best course of action based on history of actions, goals, and interactions with other agents [17]. Similar to the way of humans cooperate with others to achieve a goal, these agents need to learn to cooperate and work together based on observations to achieve a goal in an environment. In the Hanabi, players must determine the best action to take by considering actions of other players without direct communication. It is important to build a system that can control the actions of Hanabi game players to encourage cooperation between players. Autonomous and intelligent agents is a subfield of symbolic AI that includes multiple techniques such as agent simulation, multi-agent planning, negotiation protocols and so on [18].

Normative system involves applying some constraints on the ways an agent behaves based on the expectation that by applying these constraints, some advantageous objective will become visible [19]. In 1992 Shoham and Tennenholtz [17] presented a model of applying social laws in computational systems. The authors argued that social laws should be imposed as constraints in a system. This approach narrows down the list of available actions to be taken by an agent. This idea was further expanded by Van der Hoek in [20] that utilised Alternating-time Temporal Logic (ATL) introduced by [21] to understand and convey the social laws in MAS. These methods assumes that the norms are static. If an action is not allowed in a state of a system, then it will continue to be disallowed when the same state is again presented [24]. A problem with implementing this approach In Hanabi is the large state space involved especially as the number of players increases. Manually fixing the actions to take at each state individually would increase the time complexity as the algorithm would have to iterate through a large state space to find the optimal action.

The introduction of normative systems is closely related to deontic logics. This modal logic is used to detect norms in the form of permissions, obligations, and prohibitions [22]. C.E. Alchourrón in 1969 introduced the concept of normative logic to recognize logic of normative propositions [23]. The author justified that normative logic enables categorization of characteristics of normative systems such as being complete and consistent [23]. Hence, the use of deontic logic enables specification of human behaviours and norms in systems. However, a problem with this approach in Hanabi is the high computational complexity which leads to difficulty in implementation of the logics. Similar to the ATL model, the large state space in Hanabi cause it to be time-consuming and computationally expensive to implement logics in for how each player should behave. Using deontic logics, instead of learning to

play the game of Hanabi, the method of playing the game is explicitly coded into the game playing algorithm.

## 2.3 Markov Decision Process and Partially Observable Markov Decision Process

To build a multi-player game of Hanabi, it is important to determine how to model the game. Markov Decision Process (MDP) is used in a stochastic environment as an optimization model for sequential decision making. MDP was first introduced by Richard Bellman in 1954 in his paper, the Theory of Dynamic Programming [25]. In 1953, L.S. Shapley [26] introduced stochastic games for 2 player settings where the game is repeated with transition probabilities controlled by 2 players. In 1957, Bellman used dynamic programming to find optimal policies and value functions for solving optimal control problem. In 1960, Ronald A. Howard introduced policy iteration method for MDP which had the advantage of converging with fewer iterations.

The Markov property states that future is independent of the past given the present. Markov process follows the Markov property in which the probability of transitioning to the next state only depends on the previous state. Given an agent, at each timestep, the agent is at a state s. Agent transition from state s to s' by taking action a with a given probability and receives a reward. This process has the Markov property as the probability of transitioning into next state s' only depends on current state s and action a and does not take into account previous states and actions. MDP assumes that full observability of the state space, however this is not the case in Hanabi game where the state space is partially observable as agents do not have information on their own cards and the cards in the deck.

Partially Observable Markov Decision Processes (POMDP) is used in a stochastic environment as an optimization model for sequential decision making with incomplete information. In 1965, Astrom described the problem on Markov processes with incomplete state information and discrete state space can be solved using dynamic programming [27]. POMDP was used in operational research to solve optimal control problem and was later introduced in artificial intelligence by Kaelbling et. al in [28] to solve the problem of robot navigation and other real-world tasks. POMDP was more suitable for Hanabi as in Hanabi, players only receive observation from the environment and are unable to view the full state of environment.

POMDP is a generalization of MDP with the exception that in POMDP, agent does not have complete information of the environment. Besides the issue of partial observability, POMDP is MDP problem with set of states, actions transition probabilities and rewards. In each timestep, agent receives

observations based on its actions and the next state and the agent is unable to directly observe the state. The observations provide hints on the state and can depend in deterministic or probabilistic manner on the state. Hence, agent would have to keep track of past observations in order to infer what the current state could be. The goal of the agent in POMDP is same as it is in MDP setting, which is to maximize the expected discounted future reward. The POMDP framework currently only assumes a single agent in the environment. In this case, POMDP needs to be extended to the multi-agent game of Hanabi.

## 2.4 Types of Games

Game theory is a theoretical framework that studies interaction between rational agents. The founding and development of game theory could be attributed to John von Neumann and Oskar Morgenstern who published the book, Theory of Games and Economic Behavior which discussed the use of theory of games as model for economics. The field of game theory was further by the works of John F. Nash Jr in 1949 in [29] which introduced the existence of equilibrium strategies in non-cooperative multi-player games. The are many different branches of game theories.

In basic form of MDP and POMDP, a single agent makes actions and obtain rewards. This is different in multi-agent case where each agent may have their own goals and wants to maximise their own reward by minimizing other agents' reward. Hence, the reward an agent receives depends on the actions of other agents. Normal form game is usually represented as matrix games with n number of players. Each player has finite set of strategies or possible actions and payoffs. Each player executes actions simultaneously and receives rewards based on joint action.

Normal form games in general do not consider the environment dynamics which can change depending on the actions taken by the agents. In Hanabi, agents have a set of different possible actions they could choose from, and each action would transition them to different states. Stochastic games introduced by L.S. Shapley considers the state of the environment in terms of transition probabilities. In the case of Hanabi, each player chooses an action every timesteps according to the player's policy. The actions chosen by agent and the current state determines the rewards received by each agent and the transition probability associated to the new environment state [30]. The policies of the Hanabi game playing agents depend on history of states and actions in the past timesteps, and this is fully observable to all agents. Stochastic games have Markov property as the probability of transitioning to the next state depends only on the current state and action. Hence, MDP is general form of stochastic games with one player.

Besides the environment dynamics, the partial observability of the states in Hanabi games also needed to be considered. Partially observable stochastic games (POSG) are games which agents are not able to fully observe the state and the actions of all agents as in the case of Hanabi. Instead, agents receive observations and rewards from taking actions and these observations may not contain the state information as well as actions of other agents. POMDP is general form of POSG with only one agent. The reward function associated to states and actions determines if agents are cooperating or competing to receive rewards [31]. In Hanabi, players cooperate and receive a final same reward for all players. Decentralized partially observable Markov decision process (DEC-POMDP) is a case of POSG in which agents have the same reward function [31] and can be applied to Hanabi.

## 2.5 Reinforcement Learning in games

Games provide a challenging domain to test different algorithms and their performances. This is because games are designed to provide a challenge and engage with humans. Application of different RL algorithms and strategies in games would enable researchers to gain insight into how humans think and form decisions. The different types of RL methods that can be used in Hanabi include value-based methods that learn and optimizes value functions and policy-based methods that tries to learn the optimal policy directly. These different types of RL algorithms and methods have been introduced in games to produce state-of-art performances that could rival human players. Hence, the different approaches employed in playing different types of games can be reviewed to study various game playing methods that can be extended to Hanabi.

The game of Go has been a challenge in the AI domain due to large search space involved to find optimal strategies. AlphaGo was an algorithm developed by DeepMind that managed to beat managed to beat world champion Go player Lee Sedol. AlphaGo used 'policy networks' to choose actions and 'value networks' to learn board positions [32]. The policy network was first trained by supervised learning using human data and then improved using policy gradient reinforcement learning through self-play. Self-play was done by playing games using current policy networks and policy networks from previous iterations to prevent overfitting to current policy [32]. The value networks were then trained using neural networks with similar architecture to policy network to output single prediction. AlphaGo then used Monte Carlo Tree Search algorithm together with the policy and value networks to choose actions by lookahead search [32].

Classical Q-Learning is a model-free reinforcement learning method that uses Q-table to store all state-action values of agents. In Q-learning, agents try to learn the value of an action at a given state. The advanced of neural network enabled the introduction of Deep-Q Networks (DQN) by Mnih et al. in

[33] to approximate the state-value function using neural networks which improved the scalability of the algorithm. Rainbow was work that combined new improvements made to DQN and built to play the game of Atari in which it achieved state-of-the-art performance on Atari 2600 benchmark [34]. Extensions added to DQN includes using double Q-learning to improve overestimation bias, prioritized replay to sample import transitions, dueling networks, multi-step learning, distributional DQN and noisy nets [34]. The resulting algorithm achieved record performance of 223% in no-ops regime and 153% in the human regime [34].

Past no-limit Texas hold'em poker game playing algorithms were focused on playing the game of poker in a two-player setting and built to approximate Nash equilibrium strategy rather than designed to beat the other player by determining the player's flaws. A problem with using Nash equilibrium strategy is that there are not effective algorithms for finding one in two-player non-zero-sum games [35]. Pluribus is an algorithm developed on play six-player game of poker that managed to beat human players by an average of 32 milli-big blinds per game for 10000 hands played [35]. Pluribus used self-play to learn against copies of itself to find optimal strategies as self-play in practice has proved to do well in certain games with more than two players [35]. The self-play algorithm produced the blueprint strategy which was further improved during actual game using tree search to find better strategies. To reduce game complexity, action and information abstractions were introduced where certain actions or information that were similar were bucketed together.

Building an AI algorithm that can imitate and play in the manner of human players was a challenge tackled by Cicero in the game of Diplomacy where seven players interact with each other and assume the role of European countries before World War 1 and competes to poses supply centers. Cicero used a dialogue module together with strategic reasoning module and filtering process to reject incoherent messages [36]. The dialogue model was trained using a pretrained language model and human players' data of Diplomacy game. The human players' data was augmented with intents in terms of actions and used to train dialogue model to imitate human behaviour through behaviour cloning (BC). Then, Cicero uses strategic reasoning module that predict other players' policies to select intent actions for current turn [36]. The KL-regularized planning algorithm, PiKL was used to model policies of other players as using pure BC algorithm alone was not stable. Using these improvements, Cicero was ranked in the top 10% of players who played competed in two games and above [36].

## 2.6 Imitation Learning

Imitation learning is a technique in which an agent is trained to carry out tasks from demonstrations by studying how actions are mapped to observations [37]. It allows for an agent to learn complex tasks

based upon low amount of expert understanding of the tasks. Early works in imitation learning can be traced back to [38] where Autonomous Land Vehicle in a Neural Network (ALVINN) network was used to find direction a vehicle should travel based on camera images. In imitation learning (IL), behaviour cloning, (BC) and offline reinforcement learning algorithms are mainly used to learn from expert demonstrations. Imitation learning can be used in Hanabi to build a norm aware algorithm that can learn norms present in data.

Offline RL algorithms use existing datasets to generate policies that can generalize across different scenarios without the high cost associated with online data collection [39]. During training process of agent, given a large enough dataset from expert, an optimal policy can be inferred. A problem with this approach is that this technique cannot learn effectively from offline data without additional online interactions [40]. This is due to issue of distributional shift when the agent is trained under one distribution but evaluated under different distribution when different states are visited under a new policy [40].

A well-known approach for offline RL, which is Inverse reinforcement learning derives the reward function of agent given its observations [41]. On the other hand, BC is more commonly used in imitation learning as it allows for direct mappings from observations to actions by leveraging data from experts [42]. In BC, supervised learning is used to derive a policy from data. Compared to learning direct mapping, deriving a reward function incur higher computational cost to extract reward function and resolve the issue of sparse reward [42]. Since there are limitations of computational resources available for this project, the BC technique was favoured over the inverse RL and the norm aware algorithm can be built using supervised learning.

## 2.7 Deep multi-agent reinforcement learning

Multi-agent deep reinforcement learning is a combination of using deep neural networks with reinforcement learning involving more than one agent. MARL is different from single agent RL as the state of environment is determined by joint actions of all agents acting in the environment [43]. A challenge in this domain is the issue of curse of dimensionality as the number of agents in the environment increases cause an increase in state-action space [43]. There are many different algorithms that are commonly used in DMARL. These algorithms can be studied to determine the most suitable algorithm to build Hanabi game playing agents.

DQN is a value-based method that have been widely used in winning strategies of games such as Atari. The use of function approximation in RL give rise in instability in training and this is mitigated in DQN

using target networks and experience replay [44]. In [50], the Rainbow DQN was adapted to play the full version of Hanabi game with various number of players to obtain a score of 20.64 in the 2-player setting. The algorithm has similar features as in the original Rainbow DQN in the Atari game and was trained for 100 million steps for seven days.

Policy based methods on the other hand tries to learn the optimal policy by learning from the environment interactions. Policy search is done by directly searching for policies using gradient descent methods [44]. The REINFORCE algorithm introduced by Ronald J. Williams is an early version of policy gradient method. The algorithm uses Monte-Carlo sampling technique to randomly sample trajectories and update policy parameters. A problem with policy gradient methods is that gradient updates have high variance causing slower convergence [45]. This problem is made worse in multi-agent settings since the reward obtained by an agent depends on actions of other agents and when this is not considered in the agent's optimization process, the variance of gradients will increase further [46].

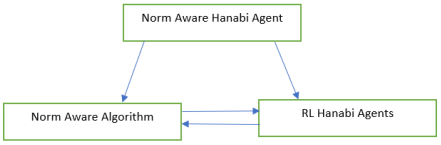
Due to high variance when implementing the classic policy gradient method, a baseline function was introduced which is subtracted from the cumulative rewards during gradient calculation. The actor-critic algorithm combines value-based methods and policy-based methods. The actor outputs the probability distributions over actions and provide the possible list of actions while the critic evaluates the current policy produced by the actor by predicting the expected return associated with each action [47]. There are many different variants of algorithms that uses actor-critic method such as Deterministic Policy Gradient algorithms introduced in [48]. In [50], actor-critic-Hanabi agent was introduced which utilized asynchronous advantage actor-critic algorithm described in [51]. However, the training time was longer compared to the Rainbow DQN Hanabi agent with final score of 22.73.

As reviewed in section 2.5, self-play is a common method utilized by state-of-the-art algorithms to train MARL agents. Hence, the RL Hanabi agents will be trained through self-play. In terms of learning, MARL algorithms use a centralised training framework with decentralized execution (CTDE) where agents learn a single policy but takes actions based on local observations or decentralised training with decentralized execution where each agent learns their own policy and takes actions based on local observations [49]. Centralized training is a well-utilized method for learning, leading to higher stability in training. However, centralized training assumes independence of agents' policies and limits agents from using global training information leading to suboptimal results [49]. Hence, the method of training Hanabi agents would be through decentralized training.

## 2.8 Conclusion

The process of completing the literature review enabled the building of game playing Hanabi algorithm that can play games according to norms by determining the important aspects of a norm aware agent. Based on literature review, social norms are the most suitable type of norms to be incorporated in Hanabi as it involves agents cooperating with each other to achieve common reward. There are many types of actions that can be considered as social norms in Hanabi, and these will be defined in the next section. As there are no current method that plays the game of Hanabi while being aware of norms, this research would involve the building of norm aware agents that combines social norms and multi-agent reinforcement learning. Incorporating norm compliance multi-agent AI systems would enable building AI systems that can gain trust of humans despite not taking optimal actions. For example, in the case of self-driving cars, the AI system may choose to follow the normative behaviour of some drivers and choose to go slowly even though this may cause the self-driving car to arrive later at a destination. By mimicking the pace of the driver, the AI system can gain the trust of human drivers and work more efficiently with humans. As conclusion after the literature review, the primary aim and the first objective have been completed.

# Chapter 3 Methodology




The methods of building a norm aware agent were broken down into 3 main parts, building a norm aware algorithm, developing an agent that plays Hanabi by reinforcement learning and combining these 2 processes to produce a final agent that can play Hanabi while being aware of norms. The final implementation and analysis would help to fulfil the secondary aims and the second, third and fourth objectives.

## 3.1 Game Profile of Hanabi

Instead of the full version of Hanabi, this research focused on a smaller version of Hanabi with 2 firework colours instead of 5 firework colours available in the full version of game. The number of cards in each players' hand at any time will be two and there will be three information tokens and one life token in the game. This is different from the full version of the game in which each player can have 4 to 5 cards on hand, and there will be 8 information tokens and 3 life tokens. The game structure of Hanabi can be defined in a tuple $S = (p, f, n, h, d, i, l)$ with the following components:

- $p \in \mathbb{N}$ is the number of players.
- $f \in \mathbb{N}$ is the number of firework colours.
- $n \in \mathbb{N}$ is the number of ranks of each firework colours.
- $h \in \mathbb{N}$ is the hand size of each player.
- $d \in \mathbb{N}$ is the deck size of cards.
- $i \in \mathbb{N}$ is the number of information tokens.
- $l \in \mathbb{N}$ is the number of life tokens.

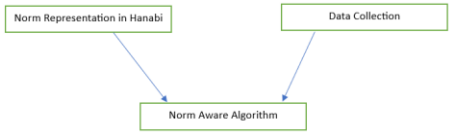
Based on the above components, the tuple $S = (2, 2, 5, 2, 20, 3, 1)$ was used for this research.

The game of Hanabi is imperfect information game as the players are not able to observe their own cards as well as the cards in deck. Hence, there is partial observability in states of the environment. Throughout this research, a game in Hanabi was called as an episode. As discussed in the literature review, Hanabi can be modelled as Dec-POMDP problem. Here, the environment of the game can be modelled as tuple $J = (P, S, A^i, O^i, T, R, \Omega, h, \Upsilon)$ with components as below:

- $P$ is the set of agents in the environment.
- $S$ is the set of states in the environment.
- $A^i$ is the set of possible actions for each agent $i$. The set of joint actions of all agents is $A = \times_i A^i$. Since in Hanabi, agents play the game in turns rather than simultaneously, only a single action $a_t^i \in A^i$ is taken by agent $i$ at any timestep t.
- $O^i$ is the set of observations for each agent i. The set of joint actions of all agents is $O = \times_i O^i$ . At each timestep, each agent $i$ will have their own observation, $o_t^i$.
- $T$ is the state transition function where $T : S \times A \rightarrow P(S)$. This defines the probability of transitioning from state $s \in S$ by taking action $a \in A$ to transition to the next state $s' \in S$ for each agent i.
- $R$ is the reward function where $R : S \times A \rightarrow \mathbb{R}$ gives the immediate reward when agent $i$ takes action $a \in A$ in state $s \in S$. All agents have the same reward function.
- $\Omega$ is the observation function where $\Omega : S\ X\ A \rightarrow O$.
- $h$ is the horizon. In the case of Hanabi, the horizon is infinite as there is not fixed number of timesteps for the game to be played.
- $\Upsilon$ is the discount factor in the range [0,1].

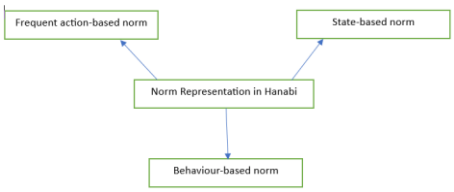
For each episode, the sequence of timesteps will be $(s_0, o_0, a_0, r_1, s_1, o_1, a_1, r_2, \dots)$. The state $s_t \in S$ captures the full configuration of the game at timestep t. At each state, each agent $i \in P$ receives observation $o_t^i$ at each timestep. Assuming a game of 2 agents, i and j, let $\boldsymbol{h_i}$ = hand information of agent $i \in P$ which contains the colours and ranks of cards on hand, $\boldsymbol{f}$ = fireworks formed which contains the firework colours and ranks formed, $\boldsymbol{d}$ = remaining deck size where d $\leq$ 16, $\boldsymbol{u}$ = discard pile information, $\boldsymbol{p}$ = ID of previous agent $i \in P$, $\boldsymbol{K}$ = information revealed in last action, $q$ = all information revealed about player's cards at hand which includes the rank and colour of cards, c = vector of which cards in hand were revealed, b = position of card that was played or discarded, and $v$ = the colour and rank of card that was last played. $\boldsymbol{K}$ contains previous player, i's action type or $a_{t-1}^i$, target card from previous action, colour revealed in last action and rank revealed in last action. This observation contains $o_t^j = (\boldsymbol{h_i}, \boldsymbol{f}, \boldsymbol{d}, \boldsymbol{u}, \boldsymbol{p}, \boldsymbol{K}, q, c, b, v, i, l)$. Based on this observation, an agent j selects action $a_t^j$. Hence, each $a_t^j \in A^j$ is dependent on $o_t^j$.

## 3.2 Norms aware algorithm



To build a norm aware agent, the types of norms in Hanabi were first be defined. Then, data collection on past instance of Hanabi game was done and then labelled with the corresponding norms. Finally, a novel norm aware algorithm was developed using the data collected based on the norms.

### 3.2.1 Norms representation in Hanabi




To build a norm aware agent that imitates human behaviour and norms, different norms present in in human society were extracted and incorporated into the game data to be learnt by the norm aware algorithm. The primary type of norm in human society that was used for this purpose was social norms as this type of norm is prevalent in Hanabi players' interactions with each other during each game episode. During players' interactions, certain action may be more common in some episodes of the game.

Social norms are also prevalent in more general aspect of the players' interactions where the most frequent actions as whole in the game are taken by the players. Finally, social norms in Hanabi are also present in situations where a player has a common response to a move played by another player. In this context, norms are a way of generalizing what to expect from certain game observations. These norms were first defined and then incorporated into the Hanabi game data. The formulation and addition of these norms in the game of Hanabi was novel contribution of this project.

**Frequent action-based norm**

Humans in general follow certain social order where the most frequent action that are commonly taken in a society are adapted as norms, for example when an individual enters a new city, they may take the most frequent action taken as whole in the new community. This can be seen as most basic form of social norm where individuals follow the actions of other individuals, resulting in a single most frequent action commonly taken by a group of individuals in a social setting. Even in multi-agent systems, the agents sometimes only do one-step lookahead search to find the most frequent action at a given state. This action may have high probability given the current state of agents and hence causing that action to be executed frequently. This type of social norm can be considered as frequency-based norm and can be extended to the Hanabi game.

In Hanabi, a player can provide a hint of rank or colour given there are information tokens left. Here, the action of hinting rank can be defined as a norm compared to hinting colour when a player chooses to hint a card. In a small version of Hanabi, there are only 2 colours of firework cards but there are 5 ranks total for each firework. Intuitively, the hint of rank would reveal more information about the card to player holding the card. Hinting rank enables players to communicate with other players if they hold a card that is relevant to building the firework. Hence, given an agent $i \in P$, $a^i_{t,1} = hint\ rank, a^i_{t,2} = hint\ colour$ and $a^i_{t,1}, a^i_{t,2} \in A^i\ with\ \Pr(a^i_{t,k}) = probability\ of\ action\ k\ at\ timestep\ t\ by\ agent\ i, \Pr(a^i_{t,1}) > \Pr(a^i_{t,2})\ \forall\ i, t \in h$. For all agents in $P$ with available actions hint rank and hint colour, the agent has higher probability of choosing the action of hinting rank than hinting colour across all timesteps in a horizon.

**State-based norm**

The type of norms individuals adhere to also can be more specified to include a frequent action at a specific state. For example, only driving slowly at a specific road due to constant high traffic at that road. This type of norm is an extension of the frequent action-based norm to only include actions made at a specific state individuals are in and hence are labelled as state-based norms. The frequency-based norm alone lacks in complexity as it only considers the immediate possible action without considering the current state of agents. Hence, adding the state of the environment adds more context to norm definition. Since players in Hanabi does not have full view on the current state of system, the observation of each player will be used when defining the state-based norm in the context of Hanabi game. State-based norms can be extended to the case of Hanabi in the first timestep of an episode.

During the first timestep, players can play a card or hint a card. Here, the action of discarding a card cannot be taken as no information token has yet been removed. Since the player in the first turn has no information on the cards on hand but are able to view the cards on other players' hand, it is a more intuitive and logical choice to provide hint especially since playing the wrong card could result in players losing a life token. The norm can be specified further by including the type of hint provided by a player during the first timestep. Here, the player could provide a hint of rank 1 if the player holds the cards with rank 1. This is because the firework for each colour must be played in order from rank 1 to 5. Hence, providing a rank hint of 1 on the first round if the other player has that rank of card enables that player to then play that card and form the first rank in the firework. Given an agent $i \in P$, at timestep t = 0 observes $o^i_0 = card\ of\ rank\ 1$, then action $a^i_0 = hint\ rank\ s.t\ rank = 1\ and\ a^i_0 \in A^i\ \forall\ i$. An agent i that observes a card of rank 1 at timestep 0 will take the action of providing a rank hint of 1 at timestep 0. This is true for all agents.

**Behaviour-based norm**

Individuals also follow a pattern of normative behaviours in some situations, for example an individual waving at a person they recognize and that person waving back at the individual. This type of normative behaviour can be labelled as behaviour-based norm and is an extension of the frequency-based norm. While the frequency-based norm only focused on single action, here a chain of actions was considered when defining normative behaviour. When acting based on frequency-based norms, agents were short-sighted as they only consider the immediate action but when acting based on behaviour-based norms, agents will be long-sighted as agents will consider a string of actions for a fixed timesteps. In the 2-player game of Hanabi, normative behaviour can be observed when player 1 provides hint of colour and the player 2 on the next turn will discard a card. Intuitively, assuming the ranks of the cards on hand is already known by player 2, then player 1 could provide colour hint to player 2 to reveal more information about card on hand. Then, player 2 may choose to discard a card to gain back information token removed when player 1 provided the colour hint. Hence, given an agent $i \in P$, $a_{t-1}^{i}$ = hint colour, $a_{t-1}^{i} \in A^{i}$ then $a_{t}^{j}$ = discard card where $a_{t}^{j} \in A^{j}$. Here, it is implicitly assumed that information token is available when providing the colour hint. For all agents, if the action of the previous agent was hint colour, then the action of the current agent will be discard card.

### 3.2.2 Data collection

In order to incorporate Hanabi game data with norms, first the game data of past Hanabi games must be collected. Here, data of human players playing Hanabi can be used to identify normative behaviours. However, due to lack of human data, data was instead generated by building a multi-agent reinforcement learning using Deep Q-Networks (DQN) with action masking. The details of this DQN agent will be discussed in section 3.3. Two Hanabi agents will first be trained to play Hanabi using DQN. Then, the performance of the trained agents will be evaluated in an online Hanabi reinforcement learning environment and the data generated was collected for a total of 50000 episodes. The information collected in the data is the observation, reward, actions, player information and timesteps.

### 3.2.3 Norm aware algorithm

Once the data was collected, the data was annotated with the frequent action-based, state-based and behaviour-based norms. Then, the data was divided into train and test set and a deep neural network algorithm was fit onto the training set. Here, supervised learning was used to learn the actions of agents and presence or absence of each of the 3 norms for each observation. Finally, the results of

this algorithm were analysed by evaluating the norm aware algorithm on unseen test set and the accuracy was noted on the predictions of actions and the 3 different types of norms.

## 3.3 Multi-agent Reinforcement Learning for Hanabi

To build 2-player agents that can learn to play Hanabi, reinforcement learning methods were considered. There are a few pre-existing reinforcement learning methods that had been implemented in full version of Hanabi game to build Hanabi learning agents. In [50], Rainbow agent using Rainbow DQN [34] was introduced as well as an actor-critic Hanabi agent using asynchronous advantage actor-critic algorithms [51] which were both trained through self-play. In [52], Bayesian Action Decoder method which used Bayesian update to obtain public belief of the environment conditioning on actions of agents was trained using self-play to play Hanabi and achieved a score of 24.174 on an average of 100000 games. The Rainbow agent and the actor-critic Hanabi agent (ACHA) both achieved an average score of 20.64 and 22.73 each respectively [50].

The main limitations for this research project are computation time and computing resource limitations. Hence, the selected must be able to find optimal policies for learning agents with limited amount of time and CPU and GPU resources available. To build the first DQN agent for data collection, a variant of the Rainbow agent was used where the agent was trained for 4.2 million timesteps without distributional DQN and Dueling architecture. Here, action masking was also used to mask out invalid actions during training and evaluation. Action masking was used to avoid repeatedly sampling invalid actions from discrete action space and this method masks out invalid actions and only selects valid actions. According to [53], modifying action space makes learning easier.

For the RL Hanabi agent, the Rainbow DQN algorithm was reimplemented with hyperparameters as in [50]. The agent was trained for 20 million timesteps. Action masking was not used as the original implementation did not make use of action masking.

## 3.4 Norm-aware Hanabi agent

To build multi-agent Hanabi players that can play with reinforcement learning while being aware of norms, both the final norm aware algorithm and trained RL agents were used in an online environment of Hanabi game during evaluation. During action selection, a lambda value was used to select action of agents using either the output from norm aware algorithm or the trained RL algorithm. The usage of this lambda value was inspired from the works of building Cicero for the game of Diplomacy in [36]

that was discussed in the literature review. Here, piKL algorithm was used to model the policies of players.

In Cicero, behaviour cloning was done using supervised learning of human data of Diplomacy game. However, pure behaviour cloning approach was found to be unstable hence was combined with output of a strategic planning model. piKL assumed each player $i$ is both trying to maximise expected value of their policy $\pi_i$ and minimize the Kullback-Leibler divergence between $\pi_i$ and the behaviour cloning policy [36]. A parameter λ was used to select action based on trade-offs between these 2 policies. Inspired by this work, a parameter λ was used to select actions of agents using norm aware algorithm with probabilities of 0% 50% and 100% and using the RL algorithm otherwise during evaluation for a total of 10000 games. The results in terms of the mean episode reward for both agents received for the 10000 games were evaluated and recorded.

## 3.5 Conclusion

This chapter focused on the methods utilized to build a norm aware Hanabi game-playing agent. Hence the main contributions of this research were firstly, the defining of 3 different types of norms and how they can be extended to the game of Hanabi. This is a point of novelty in this research project. Another contribution was building a norm aware Hanabi game playing agent that can play Hanabi using a norm aware algorithm and RL algorithm based on a lambda value. Depending on the lambda value, the final Hanabi game-playing agent may select the best action that maximises their reward of select action that are norms.

# Chapter 4 Implementation and Analysis

In this section, based on the literature review on norms, normative systems and reinforcement learning methods, and the method of application of a norm-aware multi-agent Hanabi agent, the norm-aware multi-agent Hanabi playing algorithm was constructed. The method of constructing this multi-agent system was based on the methodology described in chapter 3. The Hanabi environment used will be from the PettingZoo library and the RL agents will be built and tuned using Ray framework. The norm learning algorithm was built using neural networks with Pytorch. The implementation of the algorithms was done using Google Collaboratory notebook and Kaggle notebooks as the platforms offered free CPU and GPU.

PettingZoo is a pythonic interface to solve many different MARL problems. PettingZoo provides many different reference environments to solve MARL problems such as chess, Hanabi and mechanics to create custom environments. Interactions with the environment is similar to that in Gymnasium. The environment takes in input the parameters for number of firework colours, number of ranks, number of players, the hand size of each player, the number of information tokens, the number of life tokens and the observation type. This Hanabi interface is a wrapping of the Hanabi learning environment from DeepMind.

For the Hanabi environment in PettingZoo, the observation space of each player consists of a dictionary containing the observation and the action mask. Observation is in the vector of size 171 and in this vector, game information is encoded in binary form. The information encoded are the vector of card 1 and 2 in other player's hand, the unary encoding of remaining deck size, the vector of red and yellow firework, unary encoding of remaining information tokens, unary encoding of remaining life tokens, thermometer encoding of discard pile, one-hot encoding of previous player's ID, vector of previous player's action type, vector of target from previous action, vector of colour revealed in last action, vector of rank revealed in last action, vector of which cards in hand were revealed, position of card that was played or removed, vector of last card that was played and revealed information of current player's first and second card.

Finally, the action mask is a binary vector with a value 1 indicating if the action is legal to be taken. The size of vector is 11. Each value in the vector indicates, in the following order; discard card at position 0, discard card at position 1, play card at position 0, play card at position 1, reveal red cards for player, reveal yellow card for player, reveal rank 1 card to player, reveal rank 2 cards to player, reveal rank 3 cards to player, reveal rank 4 cards to player and reveal rank 5 cards to player. The action value is counted from position 1, for example action value of 5 is reveal red cards for player. The

reward for each step was calculated as the change in score in terms of fireworks made from the previous step. If illegal action was taken by the agent, then the game terminates with a negative reward given to the agent that took the illegal action and reward of 0 for the other agents.

Ray is unified framework for distributed computing that offers multi machine learning libraries to solve machine learning workloads. Ray was used for this research project as it was flexible to be implemented. It provided frameworks and tools to implement RL algorithms for multi-agent training as well as tools for hyperparameters tuning. For this project, RLlib and Tune libraries were used to implement the algorithms and train and tune the algorithms respectively. Tune is a Python library for hyperparameter tuning at scale and it was used for hyperparameter tuning and training of RL agents as it allows parallelism where hyperparameter tuning can be distributed across multiple CPUs.

RLlib was used to implement RL agents. The algorithm uses policies to select actions in the Hanabi environment. Given a policy, sample batches are produced from rollouts through environment. Sample batches are collected separately for each policy and wrapped up together in MultiAgentBatch. For the 2 agents, RLlib collect data from the environment based on agents' policies and make sure the policies' weights are in sync for each agents' policies. Each algorithm class in RLlib is managed by its respective AlgorithmConfig. For example, the DQN class is managed by DQNConfig that is used to specify the configuration of the DQN algorithm.

In the following section, the implementation details and the analysis of the norm learning algorithm were first discussed. Then, the implementation details of the RL agents were discussed and the analysis on the performance of these algorithms was done. Finally, these processes will be used to describe the implementation detail of the final agents that plays Hanabi using RL while being norm aware. By completing this section, the primary and secondary aim of building a norm aware algorithm and building a reinforcement learning agents that can play Hanabi while being aware of norm can be fulfilled. This also directly helps in fulfilling the last 3 objectives.

## 4.1 Norm aware algorithm

To build a norm aware algorithm, first DQN was used to train 2 agents to learn to play Hanabi and the data from these 2 agents were used to build the norm aware algorithm. The details of this DQN algorithm will be discussed in section 4.2.1 and 4.2.2 of this chapter. Using the data, norms were first incorporated into this data and the distributions of norms in this data were plotted and analysed. Then, the details of building a norm aware algorithm based on this data were discussed and the results in terms of the accuracy of the algorithm were analysed.

### 4.1.1 Norms representation in data

The following section discusses the norms representation process in the data and the analysis of the data in terms of distribution of norms after the norms have been incorporated.

**Frequent action-based norm**

For frequent action-based norm, the action of hinting rank was more frequent than the action of hinting colour. Here, the columns of actions taken by agents in the data were considered, specifically, the action of hinting colour which has action values of 5 and 6 for hinting red and yellow and the action values to 7 to 11 for hinting rank 1 to 5. A new column with label "action_norm" was introduced where the rows with the column "action" with values 5 and 6 were labelled as 0 indicating it is not a norm and rows with the column "action" that has values 7 to 11 were labelled as 1 indicating it is a norm. The other rows of data that were not involved in determining the frequent action-based norm were given a label of 2. The below shows the count plot of frequent action-based norm in the data of 50000 games.

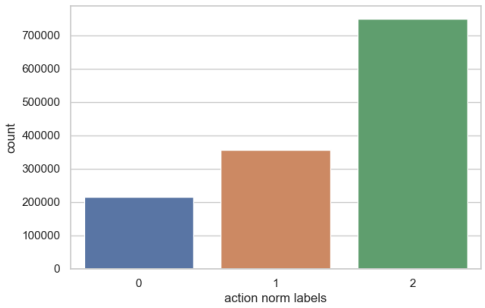



Figure 4-1: Total count of labels 0, 1 and 2 for frequent-action norm.

From the above plot in Figure 4-1, it can be observed that the action of hinting rank is more frequent than action of hinting colour as there are more action norm value one than zero. As such, hinting rank was a normative action for this dataset.

**State-based norm**

For state-based norm, it is considered a norm in Hanabi to hint rank of 1 in the first timestep of an episode of game if agents have cards on hand with rank of 1. Here, the columns of actions and observation of agents were considered. First, the rank of cards on hand were extracted from the observation vectors of all agents across episodes for all timesteps. A new column with label "state_norm" was introduced where rows with column "action" with value 1 at timestep zero and with card(S) of rank 1 in observation vector was given a label of 1 in the "state_norm" column,

indicating it is a norm. Additionally, rows with column “action” that does not have value of 1 at timestep zero were given label of zero. The other rows of data that is not involved in determining the state-based norm was given a label of 2. The below shows the plot of state-based norm in the data of 50000 games. All rows of data that state-based norms with label 1 will also have frequent-action based norms with label 1.

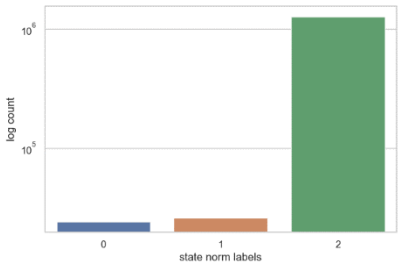



Figure 4-2: Total log count of labels 0, 1 and 2 for state-based norm.

Based on the above plot in Figure 4-2, it can be observed that the total count where the norm is present, 1 is higher than when the norm is absent, 0.

**Behaviour-based norm**

For behaviour-based norm, this form of normative behaviour was observed in Hanabi when the first player hint colour and the next player discard a card. Here, the actions of agents were analysed and a new column for the data with label “behaviour_norm” and “previous_action” which was the previous action of the past agent for each timestep were introduced. For each row of data, if the “previous_action” was hint colour and the “action” was discard card then a label of 1 was given to “behaviour_norm” column for that row of data, indicating it is a norm. For each row of data, if the “previous_action” was hint colour and the “action” was not discard card then a label of 0 was given to “behaviour_norm” column for that row of data, indicating it was not a norm. The other rows of data that were not involved in determining the behaviour-based norm were given a label of 2. The below Figure 4-3 shows the plot of state-based norm in the data of 50000 games.

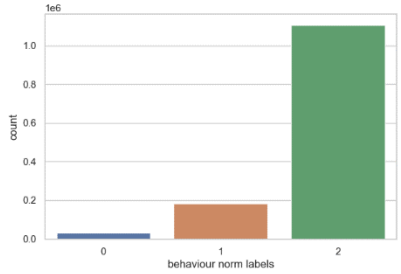



Figure 4-3: Total count of labels 0, 1 and 2 for behaviour norms.

### 4.1.2 Norms aware algorithm

The data was then used to train a norm aware algorithm using behaviour cloning. Duplicated data were removed except for duplicated rows with “state_norm” values of 1 and 0. This was because the set of possible state-based norms with labels 1 and 0 were too small and the data was not removed to prevent severe data imbalance. The data was first divided into train and test data with 30% of the data in the test set. Then, a single neural network model was used to train and build the norm aware algorithm using the training data with 4 outputs, the predictions of actions, behavioural norm, action norm and state norm. The neural network model has 2 hidden layers with sizes 256 and 512 respectively and dropouts with ReLU activation function. The learning rate was set to be 0.0001 and the model was trained for 20 epochs. The performance in terms of accuracies for both train and test set was plotted as below.

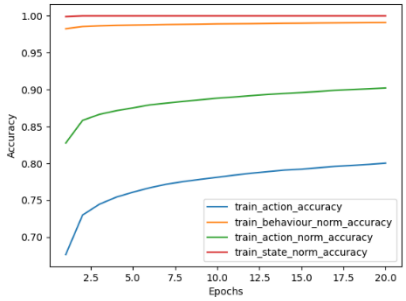



Figure 4-4: Model accuracy on train data over 20 epochs

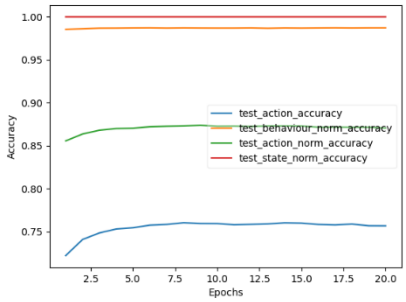



Figure 4-5: Model accuracy on test set over 20 epochs

Based on the above plots in Figure 4-4 and Figure 4-5, the training accuracy for action predictions increased up to 80% and the test accuracy for action predictions increased up to 76%. Even with dropouts introduced, there appear to be slight overfitting to the train dataset. For the training accuracy of frequent action-based norm predictions, the accuracy increased steadily up to 90% and the test accuracy of action norm predictions are 87%. The model was to be able to predict the presence of action norms with high accuracy. Additionally, the prediction accuracies of both the

training and testing data are 100% for the state based-norm predictions. This was expected as there are duplicated data for state-based norms in the dataset. There may be some data for state-based norms that were both present in train and test set and the model was able to predict them accurately. Finally, the model was also able to accurately predict the behaviour-based norm with an accuracy of close to 100% for both train and test set.

## 4.2 MARL for Hanabi

Two separate RL-based Hanabi game playing algorithms were developed, one was used for collecting data to build norm aware algorithm. The details of implementation are described in section 4.2.1 and the limitations of the algorithm are described in section 4.2.2. The second RL-based Hanabi algorithm was built as RL Hanabi agent. The implementation details are described in section 4.2.3 together with the limitations of the algorithm. In section 4.2.4, the implementation and evaluation of the final norm aware Hanabi agent is recorded.

### 4.2.1 MARL for Hanabi Data Collection

The training of Hanabi DQN agents was done using RLlib library and tuned using Tune library. The DQN agents that was trained to play Hanabi was used for data collection to build a norm aware algorithm. For the structure of the agent itself, first action masking was done to mask out invalid discrete actions for each timesteps in an episode. This is because in the Hanabi environment, some actions may not be available to the agent to be executed and action masking prevents the agents from taking these illegal actions. Kannervisto et. al [53] used different type of action space shaping masking actions and converting multi-discrete actions into discrete actions to study the effect of action shaping on the performance in video games environments. An experiment conducted in StarCraft II mini games by [53] showed improvements of performance in terms of reward received when masking unavailable actions was done.

Action masking was done by replacing the logits associated with the illegal actions with a negative infinity. In the case of DQN, the action selected by the algorithm at each timestep will the action with the highest q-value associated with a particular state. Action masking replaced the q-values of the illegal actions to negative infinity so that the illegal action was not get executed. In terms of the hyperparameters of DQN algorithm itself, both the 2 agents were trained with separate policies and do not share parameters. The algorithm consists of 2-layer multi-layer perceptron with 512 hidden units each and the batch size used was 512. The epsilon value of the epsilon greedy policy was

annealed from 0.00052 to 0 over the course of 1000 timesteps. The discount value used was 0.99 and prioritized replay buffer was used to store the q-values with buffer size of 1 million.

During each training iteration, sample batches were collected from the environment for each policy and combined into a MultiAgentBatch and stored in the replay buffer. Then, training batch in terms of the MultiAgentBatch was sampled from the replay buffer and used for training and the policy weights were updated for each agent's policy. The target network was updated every 500 sample steps and the metrics for each training iteration were recorded in terms of episode reward mean. In each training iteration, several episodes were run. For each of the 2 players, the mean of rewards obtained in these episodes were obtained for each training iteration. The sum of the means of the 2 players in each training iteration will be the episode reward mean for that iteration. In each iteration, each player has a minimum and maximum reward value which are the minimum reward obtained across episodes and the maximum reward obtained across episodes. The training was done for 4.2 million timesteps on CPU for a duration of around 20 hours. The plot of episode reward mean across iterations is as below in Figure 4-6.

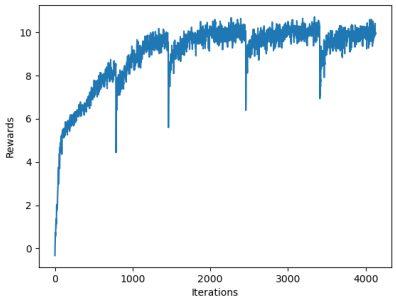



Figure 4-6: Episode reward mean for player 0 and player 1 using DQN algorithm.

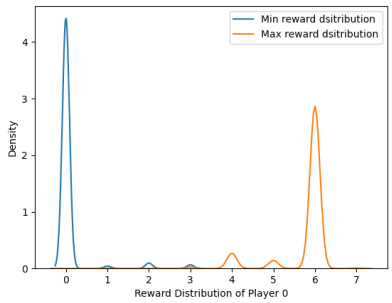



Figure 4-7: Distributions of minimum and maximum rewards across training iterations for player 0.

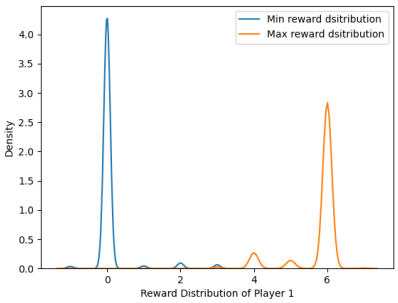



Figure 4-8: Distributions of minimum and maximum rewards across training iterations for player 1.

### 4.2.2 Limitations of Hanabi DQN agent for Data Collection

Based on the results obtained in Figure 4-6, the highest episode reward mean of the agents were 10. The maximum reward received by each agent was mostly 6 based on Figure 4-7 and Figure 4-8. The minimum reward received by each agent was mostly 0 based on Figure 4-7 and Figure 4-8. By analysing the game results, the maximum score of 6 was obtained when the fireworks for both red and yellow was built up to rank of 3. The algorithm did not learn to build fireworks of ranks of 4 and 5 for both the colours. This was because during earlier timesteps, the agents appeared to discard the cards with rank 4 and 5 hence when they have surpassed rank 3 for each firework, the agents were unable to play the cards with rank 4 and 5 as the cards have been discarded. It appears the current setting of DQN algorithm was unable to fully learn to play Hanabi to reach the full fireworks. Further hyperparameter tuning was done using Ray Tune. The hyperparameters tuned were the update frequency of the target networks, batch size, buffer size, learning rate, hidden layer size and number of steps in multi-step learning. However, the maximum mean episode reward that was able to be achieved was around 10 after which training saturates.

### 4.2.3 MARL for Norm-Aware Agent using Rainbow DQN

The Rainbow DQN Hanabi agent in [50] was reimplemented for this research with the same parameters. Action masking was not used, and the agent used a 2-layer Multi-Layer Perceptron of 512 hidden units each. The batch size was set to 32 and the epsilon value in epsilon-greedy policy to 0 over the first 1000 timesteps. The discount factor was set to be 0.99 and the algorithm used prioritized replay buffer with buffer size of 1 million. Discrete distribution was used to approximate value distributions over 51 atoms. Here, the usage of C51 algorithm enables prediction of distribution of q-values rather than a single q-value for each state-action pair. According to [54], using a distributional reinforcement learning approach leads to a more stable training and improves performance of the algorithm. The algorithm was trained for 20 million timesteps for 4 days. In the original implementation, parameter sharing was utilized between agents during training. A comparison was made with agents trained with single policy and 2 separate policies:

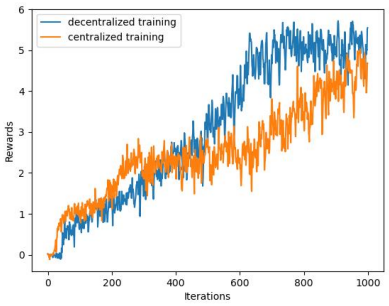



Figure 4-9: Episode reward mean across iterations for training with single policy and 2 separate policies.

Based on the results above, agents trained independently received higher reward after 400 training iterations compared to agents trained with parameter sharing. Hence, the agents were trained with separate parameters. The plot of episode reward mean of the Rainbow DQN agent is as below:

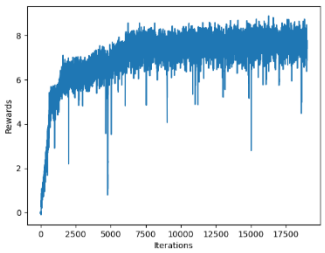



Figure 4-10: Episode reward mean for player 0 and player 1 with Rainbow DQN algorithm.

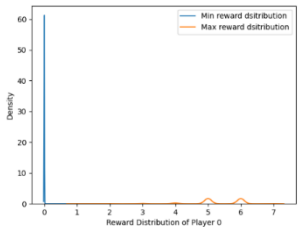



Figure 4-11: Distributions of minimum and maximum rewards across training iterations for player 0 using Rainbow DQN algorithm.

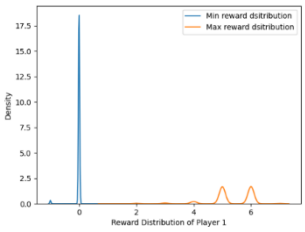



Figure 4-12: Distributions of minimum and maximum rewards across training iterations for player 1 using Rainbow DQN algorithm.

### 4.2.4 Limitations of Rainbow DQN agent

Based on the plot in Figure 4-10, the rewards increased across iterations with highest episode mean reward of around 8. However, the problem with this algorithm was that the training for improvement in performance in terms of rewards obtained was too slow. Due to time constraints, the training was

stopped after 20 million timesteps. The original Rainbow DQN Hanabi agent in [50] was run for 100 million timesteps. Based on Figure 4-11 and Figure 4-12, the minimum rewards received by players was mostly 0. The maximum rewards across training iterations was more spread out for both players compared to the DQN algorithm in 4.2.2

### 4.2.5 Norm-aware Hanabi game-playing Agent

The trained Rainbow DQN algorithm and the norm aware algorithm were used together to build a final Norm-aware Hanabi game playing agent that can play Hanabi while being aware of norms. During each timestep in an episode, the Rainbow DQN algorithm predicted the best action for current player based on the observation. Then, the observation is passed through the norm aware algorithm and presence of the 3 different types of norms based on the observation and the best action were the outputs of the algorithm. A lambda value of 1 were given to the norm aware algorithm and 0 to the Rainbow DQN algorithm. Different probabilities, 0%, 50% and 100% are given to occurrence of lambda value of 1 to vary the range in which norm aware algorithm was used during evaluation. The mean episode reward received by the agents for 1000 episodes with different lambda probabilities were recorded in Table 4-1 below.

| Probability of lambda = 1 | Mean episode reward for player 1 and 2 |
|---|---|
| 0% | 7.21 |
| 50% | 3.7 |
| 100% | 2.4 |

Table 4-1: The mean episode reward for different probabilities of lambda value 1 using Rainbow DQN agent.

When probability of lambda value 1 was 0%, the norm-aware Hanabi Agent fully played with output actions of Rainbow DQN algorithm without the norm-aware algorithm output and achieves mean episode reward of 7.21. This was similar as the mean episode reward during training of the Rainbow DQN algorithm. When the probability of lambda value 1 were 50% and 100%, the norm-aware Hanabi Agent took the actions of the norm-aware algorithm 50% and 100% of the time if the actions were norms and achieved mean episode reward of 3.7 and 2.4 respectively. The results above were as expected as when more actions were taken based on the output of norm-aware algorithm compared to the output of the Rainbow DQN algorithm for different observations, then the mean episode rewards of the players decrease. This is because, the players will sacrifice taking the best moves according to an observation in favour of taking actions according to norms. Hence, playing by normative actions may cause decrease in performance compared to just taking the best actions.

# Chapter 5 Discussion

In this research, MARL techniques and supervised learning were used to build a norm aware system. A value-based RL approach was used in building a DQN agent for data collection and final agent that plays Hanabi while being aware of norms was built. The novel contribution for this research was the definitions of different types of normative behaviours and how the norms could be applied to Hanabi game. However, this method of building a norm aware agent is still flawed and can be improved.

- Data issue to build norm aware algorithm.

Due to lack of human data available to build a norm aware algorithm, data from DQN agent was used instead. However, this DQN agent was not optimal in terms of performance as the agent was unable to achieve full firework score of 10. Hence, the norm aware algorithm that was built using supervised learning is not able to capture the full action space possibility in Hanabi as cards of rank 4 and 5 were never played. Another issue was that data from single DQN agent was used to build the norm aware algorithm and the norms captured by the algorithm is only from single agent. It is difficult to gauge the performance of the algorithm in game with real human players as the algorithm was not able to be evaluated on data of human players that may follow different strategies.

- Training time and complexity for MARL

An issue with training the RL agents was that it took a long time to train the agents. This made the process of hyperparameter tuning difficult as the algorithm needed to be trained for over 100000 timesteps in order to determine the efficiency of the tuned hyperparameter. A parameter value that gave good result in the short term sometimes caused learning to saturate after a few million timesteps causing the algorithm to perform poorly in the long term. In this case, there is time constraints involved in building RL agents that can perform well in terms of rewards received. This issue was caused by the observation and action space that were large. There was also issue on the resources used to train the algorithms. Due to the large buffer size used to train RL algorithms, the memory requirement to train the algorithms was large and the training had to be done separately every 1 million timesteps as the resources used were running out of memory.

- Sparse environment rewards

Agents received positive rewards when a card has been played correctly and a firework rank has been added. However, agents may sometimes have to wait a long time to play a card as the cards

on hand may be that of higher ranks and the cards of lower ranks are at the bottom of deck. Here, some form of reward shaping was needed in terms of different intermediate rewards agents received for different actions. This issue was prevalent when the DQN algorithm did not learn to keep the cards with rank 4 and 5 to be played in future timesteps.

- Algorithm complexity

The DQN algorithms used in this project is a value-based method. Although DQN has been widely used in many games to achieve good performance, other algorithms also can be considered. Policy-based methods, such as proximal policy optimization and multi-agent deep deterministic policy gradient can also be used as comparison to achieve better performance.

# Chapter 6 Conclusion and Future Work

In this work, supervised learning was used with reinforcement learning to build a norm aware algorithm that can identify 3 different types of norms and play Hanabi according to those norms. The task of norm identification is a complex task because the types of norms present vary depending upon situations. For example, the normative behaviours when driving on the road is different than when playing Hanabi. However, the 3 types of norms; frequent action norms, behaviour norms and state norms can still be generalised and extended to other situations by detecting actions or chains of actions that are commonly executed in those situations.

Prior works have focused on detecting and applying norms that are only present in specific task. Hence, this work introduced a novel norm aware model that can detect different types of norms and apply it with RL algorithms so that the final algorithm can play Hanabi in a more human-like manner. This helps to build an algorithm that makes decisions by considering human behaviours instead of just taking the best action at each timestep. A norm aware algorithm in this case can be integrated easily in tasks that involves cooperation with humans as it can make decisions that can be interpreted easily by humans instead of making unusual moves and decisions.

In the future work, different methods of building a norm aware algorithm could be considered. Instead of separating the process of building a norm aware algorithm and an algorithm that plays the game of Hanabi using reinforcement learning, these 2 processes can be combined. This can be done by using reinforcement learning to not only learn to play Hanabi but also learn normative actions in the game in online manner. Hence, the algorithm will output the best action and if the action is a norm. Then,

there would be no need to collect human data for Hanabi game as the learning of norms can be done online through environment interactions. Different types of algorithms also can be considered such as policy gradient methods and Bayesian methods that can model the different beliefs of agents and the performance of these algorithms can be compared. In future work, instead of just Hanabi, the idea of norm learning can also be applied in self-driving cars to learn different norms of humans when driving on the road.

# Bibliography

[1] Ho, Mark K., James MacGlashan, Amy Greenwald, Michael L. Littman, Elizabeth Hilliard, Carl Trimbach, Stephen Brawner, Josh Tenenbaum, Max Kleiman-Weiner, and Joseph L. Austerweil, "Feature-based Joint Planning and Norm Learning in Collaborative Games," In *CogSci*, 2016.

[2] B. Toghi, R. Valiente, D. Sadigh, R. Pedarsani and Y. P. Fallah, "Social Coordination and Altruism in Autonomous Driving," in IEEE Transactions on Intelligent Transportation Systems, vol. 23, no. 12, pp. 24791-24804, Dec. 2022, doi: 10.1109/TITS.2022.3207872.

[3] Shum, Michael, Max Kleiman-Weiner, Michael L. Littman, and Joshua B. Tenenbaum, "Theory of minds: Understanding behavior in groups through inverse planning," In *Proceedings of the AAAI conference on artificial intelligence*, vol. 33, no. 01, pp. 6163-6170, Jul. 2019.

[4] Ho, Mark K., Rebecca Saxe, and Fiery Cushman, "Planning with theory of mind," *Trends in Cognitive Sciences* 26, no. 11, pp. 959-971, Nov 2022.

[5] J. Togelius and G. N. Yannakakis, "General general game AI," 2016 IEEE Conference on Computational Intelligence and Games (CIG), Santorini, Greece, 2016, pp. 1-8, doi: 10.1109/CIG.2016.7860385.

[6] Shih, Andy, Stefano Ermon, and Dorsa Sadigh, "Conditional imitation learning for multi-agent games," In *2022 17th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pp. 166-175. IEEE, Mar. 2022.

[7] Köster, Raphael, Dylan Hadfield-Menell, Richard Everett, Laura Weidinger, Gillian K. Hadfield, and Joel Z. Leibo, "Spurious normativity enhances learning of compliance and enforcement behavior in artificial agents," *Proceedings of the National Academy of Sciences* 119, no. 3, p. e2106028118, Jan 2022.

[8] Nahian, Md Sultan Al, Spencer Frazier, Mark Riedl, and Brent Harrison, "Learning norms from stories: A prior for value aligned agents," In *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, pp. 124-130, Feb. 2020.

[9] Horne, Christine, and Stefanie Mollborn, "Norms: An integrated framework," *Annual Review of Sociology,* 46, pp. 467-487, Jul. 2020.

[10] Savarimuthu, Bastin Tony Roy, and Stephen Cranefield, "Norm creation, spreading and emergence: A survey of simulation models of norms in multi-agent systems," *Multiagent and Grid Systems* 7, no. 1, pp. 21-54, Jan 2011.

[11] Cranefield, Stephen, Michael Winikoff, Virginia Dignum, and Frank Dignum, "No Pizza for You: Value-based Plan Selection in BDI Agents," In *IJCAI*, pp. 178-184, Aug. 2017.

[12] Gomila, Robin, and Elizabeth Levy Paluck, "The social and psychological characteristics of norm deviants: A field study in a small cohesive university campus," *Journal of Social and Political Psychology* 8, no. 1, pp. 220-245, Feb. 2020.

[13] Bicchieri, Cristina, "Covenants without swords: Group identity, norms, and communication in social dilemmas," *Rationality and Society* 14, no. 2, pp. 192-228, May. 2002.

[14] Zhang, Wen, Yunhan Liu, Yixuan Dong, Wanna He, Shiming Yao, Ziqian Xu, and Yan Mu, "How we learn social norms: a three-stage model for social norm learning," *Frontiers in Psychology,* 14, pp. 1153809, Jun. 2023.

[15] Morgan, Thomas JH, Luke E. Rendell, Micael Ehn, William Hoppitt, and Kevin N. Laland, "The evolutionary basis of human social learning," *Proceedings of the Royal Society B: Biological Sciences* 279, no. 1729, pp. 653-662, Feb. 2012.

[16] Thøgersen, John, "Social norms and cooperation in real-life social dilemmas," *Journal of economic psychology* 29, no. 4, pp. 458-472, Aug. 2008.

[17] Shoham, Yoav, and Moshe Tennenholtz, "On the synthesis of useful social laws for artificial agent societies," In *Proceedings of the Tenth National Conference on Artificial Intelligence*, pp. 276-281, 1992.

[18] Cardoso, Rafael C., and Angelo Ferrando, "A review of agent-based programming for multi-agent systems," *Computers* 10, no. 2, pp. 16, Jan. 2021.

[19] Ågotnes, Thomas, Wiebe van der Hoek, and Michael Wooldridge, "Normative system games," In *Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems*, pp. 1-8, 2007.

[20] van Der Hoek, Wiebe, Mark Roberts, and Michael Wooldridge, "Social laws in alternating time: Effectiveness, feasibility, and synthesis," *Synthese* 156, pp. 1-19, May. 2007.

[21] Alur, Rajeev, Thomas A. Henzinger, and Orna Kupferman, "Alternating-time temporal logic," *Journal of the ACM (JACM)* 49, no. 5, pp. 672-713, Sep. 2002.

[22] Boella, Guido, and Leendert Van Der Torre, "A game-theoretic approach to normative multi-agent systems," In *Dagstuhl seminar proceedings*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2007.

[23] Alchourrón, Carlos E, "Logic of norms and logic of normative propositions," *Logique et analyse* 12, no. 47, pp. 242-268, Sep. 1969.

[24] Huang, Xiaowei, Ji Ruan, Qingliang Chen, and Kaile Su, "Normative multiagent systems: A dynamic generalization," *arXiv preprint arXiv:1604.05086*, Apr. 2016.

[25] Bellman, Richard, "Dynamic programming," *Science* 153, no. 3731, pp. 34-37, 1966.

[26] Shapley, Lloyd S, "Stochastic games," *Proceedings of the national academy of sciences* 39, no. 10, pp. 1095-1100, Oct. 1953.

[27] Åström, Karl Johan, "Optimal control of Markov processes with incomplete state information I," *Journal of mathematical analysis and applications,* 10, pp. 174-205, 1965.

[28] Kaelbling, Leslie Pack, Michael L. Littman, and Anthony R. Cassandra, "Planning and acting in partially observable stochastic domains," *Artificial intelligence* 101, no. 1-2, pp. 99-134, May. 1998.

[29] Nash Jr, John F, "Equilibrium points in n-person games," *Proceedings of the national academy of sciences* 36, no. 1, pp. 48-49, Jan. 1950.

[30] Solan, Eilon, and Nicolas Vieille, "Stochastic games," *Proceedings of the National Academy of Sciences* 112, no. 45, pp. 13743-13746, Nov. 2015.

[31] Hansen, Eric A., Daniel S. Bernstein, and Shlomo Zilberstein, "Dynamic programming for partially observable stochastic games," In *AAAI*, vol. 4, pp. 709-715. 2004.

[32] Silver, David, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser et al, "Mastering the game of Go with deep neural networks and tree search," *nature* 529, no. 7587, pp. 484-489, Jan. 2016.

[33] Mnih, Volodymyr, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves et al, "Human-level control through deep reinforcement learning," *nature* 518, no. 7540, pp. 529-533, Feb. 2015.

[34] Hessel, Matteo, Joseph Modayil, Hado Van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Azar, and David Silver, "Rainbow: Combining improvements in deep reinforcement learning," In *Proceedings of the AAAI conference on artificial intelligence*, vol. 32, no. 1, Apr. 2018.

[35] Brown, Noam, and Tuomas Sandholm, "Superhuman AI for multiplayer poker," *Science* 365, no. 6456, pp. 885-890, Aug. 2019.

[36] Meta Fundamental AI Research Diplomacy Team (FAIR)†, Anton Bakhtin, Noam Brown, Emily Dinan, Gabriele Farina, Colin Flaherty, Daniel Fried et al, "Human-level play in the game of Diplomacy by

combining language models with strategic reasoning," *Science* 378, no. 6624, pp. 1067-1074, Dec. 2022.

[37] Hussein, Ahmed, Mohamed Medhat Gaber, Eyad Elyan, and Chrisina Jayne, "Imitation learning: A survey of learning methods," *ACM Computing Surveys (CSUR)* 50, no. 2, pp. 1-35, Apr. 2017.

[38] Dean A. Pomerleau, "ALVINN: an autonomous land vehicle in a neural network", *In Proceedings of the 1st International Conference on Neural Information Processing Systems (NIPS'88)*. MIT Press, Cambridge, MA, USA, 305–313, 1988.

[39] Kumar, Aviral, Joey Hong, Anikait Singh, and Sergey Levine, "When should we prefer offline reinforcement learning over behavioral cloning?," *arXiv preprint arXiv:2204.05618*, 2022.

[40] Levine, Sergey, Aviral Kumar, George Tucker, and Justin Fu, "Offline reinforcement learning: Tutorial, review, and perspectives on open problems," *arXiv preprint arXiv:2005.01643*, 2020.

[41] Arora, Saurabh, and Prashant Doshi, "A survey of inverse reinforcement learning: Challenges, methods and progress," *Artificial Intelligence* 297, pp. 103500, 2021.

[42] B. Zheng, S. Verma, J. Zhou, I. W. Tsang and F. Chen, "Imitation Learning: Progress, Taxonomies and Challenges," in IEEE Transactions on Neural Networks and Learning Systems, 2022, doi: 10.1109/TNNLS.2022.3213246.

[43] Papoudakis, Georgios, Filippos Christianos, Arrasy Rahman, and Stefano V. Albrecht, "Dealing with non-stationarity in multi-agent deep reinforcement learning," *arXiv preprint arXiv:1906.04737*, 2019.

[44] K. Arulkumaran, M. P. Deisenroth, M. Brundage and A. A. Bharath, "Deep Reinforcement Learning: A Brief Survey," in IEEE Signal Processing Magazine, vol. 34, no. 6, pp. 26-38, Nov. 2017, doi: 10.1109/MSP.2017.2743240.

[45] Rückstiess, Thomas, Frank Sehnke, Tom Schaul, Daan Wierstra, Yi Sun, and Jürgen Schmidhuber, "Exploring parameter space in reinforcement learning," *Paladyn* 1, pp. 14-24, 2010.

[46] Lowe, Ryan, Yi I. Wu, Aviv Tamar, Jean Harb, OpenAI Pieter Abbeel, and Igor Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," *Advances in neural information processing systems* 30, 2017.

[47] I. Grondman, L. Busoniu, G. A. D. Lopes and R. Babuska, "A Survey of Actor-Critic Reinforcement Learning: Standard and Natural Policy Gradients," in *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 42, no. 6, pp. 1291-1307, Nov. 2012, doi: 10.1109/TSMCC.2012.2218595.

[48] Silver, David, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller, "Deterministic policy gradient algorithms," In *International conference on machine learning*, pp. 387-395. Pmlr, Jan. 2014.

[49] Zhou, Yihe, Shunyu Liu, Yunpeng Qing, Kaixuan Chen, Tongya Zheng, Yanhao Huang, Jie Song, and Mingli Song, "Is Centralized Training with Decentralized Execution Framework Centralized Enough for MARL?," *arXiv preprint arXiv:2305.17352*, May. 2023.

[50] Bard, Nolan, Jakob N. Foerster, Sarath Chandar, Neil Burch, Marc Lanctot, H. Francis Song, Emilio Parisotto et al, "The hanabi challenge: A new frontier for ai research," *Artificial Intelligence,* 280, p.103216, Mar. 2020.

[51] Mnih, Volodymyr, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," In *International conference on machine learning*, pp. 1928-1937. PMLR, Jun. 2016.

[52] Foerster, Jakob, Francis Song, Edward Hughes, Neil Burch, Iain Dunning, Shimon Whiteson, Matthew Botvinick, and Michael Bowling, "Bayesian action decoder for deep multi-agent reinforcement learning," In *International Conference on Machine Learning*, pp. 1942-1951. PMLR, May. 2019.

[53] A. Kanervisto, C. Scheller and V. Hautamäki, "Action Space Shaping in Deep Reinforcement Learning," 2020 IEEE Conference on Games (CoG), Osaka, Japan, 2020, pp. 479-486, doi: 10.1109/CoG47356.2020.9231687.

[54] Bellemare, Marc G., Will Dabney, and Rémi Munos, "A distributional perspective on reinforcement learning," In *International conference on machine learning*, pp. 449-458. PMLR, Jul. 2017.