

Augmenting the action space with conventions to improve multi-agent cooperation in Hanabi

F. Bredell^{1*}, H. A. Engelbrecht¹ and J. C. Schoeman¹

¹Electrical and Electronic Engineering, University of Stellenbosch,
Western Cape, South Africa.

*Corresponding author(s). E-mail(s): 20795718@sun.ac.za;
Contributing authors: hebrecht@sun.ac.za; jcschoeman@sun.ac.za;

Abstract

The card game *Hanabi* is considered a strong medium for the testing and development of multi-agent reinforcement learning (MARL) algorithms, due to its cooperative nature, partial observability, limited communication and remarkable complexity. Previous research efforts have explored the capabilities of MARL algorithms within Hanabi, focusing largely on advanced architecture design and algorithmic manipulations to achieve state-of-the-art performance for various number of cooperators. However, this often leads to complex solution strategies with high computational cost and requiring large amounts of training data. For humans to solve the Hanabi game effectively, they require the use of conventions, which often allows for a means to implicitly convey ideas or knowledge based on a predefined, and mutually agreed upon, set of “rules” or principles. Multi-agent problems containing partial observability, especially when limited communication is present, can benefit greatly from the use of implicit knowledge sharing. In this paper, we propose a novel approach to augmenting an agent’s action space using *conventions*, which act as a sequence of special cooperative actions that span over and include multiple time steps and multiple agents, requiring agents to actively opt in for it to reach fruition. These *conventions* are based on existing human conventions, and result in a significant improvement on the performance of existing techniques for self-play and cross-play for various number of cooperators within Hanabi.

Keywords: Multi-agent Reinforcement Learning, Hanabi, Agent Cooperation, Self-play, Cross-play

1 Introduction

Reinforcement learning (RL) holds promising potential to address a large variety of problems where artificial agent operation offers a significant improvement over alternative methods, such as hard-coded algorithms or solutions [1]. For certain real-world problems, single-agent operation is not optimal (or even possible) and the incorporation of multiple agents would be beneficial (or necessary). A prime example is autonomous vehicles navigating roads [2], where each vehicle is controlled by an autonomous agent and these agents must cooperate effectively to ensure road safety. Other examples of problems that benefit from multiple agents include guided drone swarms [3], mapping verbal instructions to executable actions [4], or the switching of railway lines. RL can help agents to effectively cooperate within these multi-agent scenarios by learning from past and or simulated experiences.

Unfortunately, the introduction of multiple agents into an environment typically increases the complexity of the problem exponentially. It introduces a moving target learning problem [5], since all the agents must learn simultaneously. The reason for this is that each individual agent's policy changes over time, which in turn causes a non-stationary environment. This often inhibits all the agents from developing effective policies and can lead to undesired behaviour [5]. Furthermore, multi-agent systems often contain partial observability, where the full state space is hidden from the individual agents, resulting in each agent having their own unique perspective of the problem. In the case of autonomous agents controlling vehicles, even though an agent might have access to high-definition sensors (often used in vehicle control), the environment will still contain objects outside the current agent's perspective, for example objects outside the sensor's range or viewing angle or objects obscured by other vehicles, static objects, blind corners, concealed driveways, etc.

When another agent controlling a different vehicle is introduced into the environment, the need for communication to overcome the problem of partial observability becomes apparent, since this agent will have a different viewpoint of the environment. These viewpoints will have some overlap, but will also contain important information that the agents will have to share, such as obscured objects, accidents, obstacles, pedestrians, etc., in order to safely navigate the road. Even when the agents are able to communicate, it is infeasible to share and process all the high-quality information between agents, especially in scenarios where quick reaction times are crucial. This communication problem becomes even more difficult if we consider situations where more than two vehicles must coordinate effectively, for example a large intersection, road construction, an accident that lead to congestion, or a busy city.

One of the most straightforward solutions to multi-agent reinforcement learning (MARL) is the combination of deep Q-networks (DQNs) and independent Q-learning [6], where each agent independently and simultaneously learns its own action-value function (or *Q-values*¹) while interacting with the same environment. However, this strategy does not fair well when paired with partial observability [7, 8]. Hausknecht and Stone [9] have shown that recurrent neural networks offer an improved

¹Action values (Q) refer to the value assigned to a certain action a within a given state s .

solution to MARL problems containing partial observability. These networks incorporate a built-in short-term memory over which experiences are *unrolled* (or combined) to form longer sequences. This is often combined with deep Q-learning’s feed forward neural network to produce deep recurrent Q-learning [9].

Various research efforts follow a similar path as Hausknecht and Stone, focusing on architectural advancements through the means of neural network layer manipulations, and complex algorithms to improve the estimation and assignment of action-values or policies [10–12]. This often leads to convoluted solution strategies that are difficult to implement and computationally expensive [13, 14]. An alternative solution strategy would be to reconsider the problem dynamics and discover ways of incorporating existing domain knowledge into RL algorithms. Existing domain knowledge can be incorporated into RL and MARL using reward shaping [15–17], which focuses on manipulating the reward signal to encourage certain behaviour. Alternatively, agents can use state-space augmentations, such as auxiliary tasks [18, 19], to imply or extract additional information (or features) of the problem setting. Sutton *et al.* [20] have shown that *options* offer an additional strategy for incorporating domain knowledge into RL by changing the action space of an agent. Options allow an agent to solve problems on a higher level by extending the action space to include advanced temporal actions that range over multiple time steps (also referred to as *macro-actions*).

Games are often used as a medium for testing and evaluating RL algorithms, since they incorporate real-world problems with well-defined rules and a clear metric for measuring performance, where examples include *Go* [21], *Backgammon* [22], and *Dota 2* [23]. *Hanabi* is a cooperative card game containing partial observability and limited communication, requiring players to logically reason over the intentions and actions of their cooperators, a concept known as theory of mind [24]. Human players require the development of conventions² in order to solve the problem effectively. Conventions have evolved with humanity throughout the ages and range from driving on a certain side of the road to social conventions or norms [25].

Conventions allow for additional implicit communication through actions and mutually agreed upon “rules”, thereby overcoming the communication restrictions of the communication channel. Similar to the Hanabi problem, when humans tackle the problem of controlling vehicles on roads, conventions are common practice and ensure effective cooperation. Examples include: driving on a certain side of the road³; keep left, pass right (or vice versa); who has right of passage at a T-intersection or four-way intersection (and how this changes based on the presence of pedestrians); flashing of headlights while stationary to indicate giving right of way/yielding; flashing of headlights while moving to indicate danger ahead, or be wary of potential danger ahead; flashing of hazards to indicate an emergency; a quick flash of hazards to say thank you; and many more. Furthermore, this highlights the importance of artificial agents learning existing human conventions, since there will come a time when autonomous agents must navigate these complex scenarios alongside humans, and the agents will have to

²Philosopher David Lewis defines conventions as an arbitrary, self-perpetuating solution to a recurring coordination problem [25].

³Philosopher David Lewis discusses this topic in depth, and places particular focus on how this affects other aspects of life, such as people tending to keep to the same side as their road conventions when walking in a busy street [25].

be able to convey and receive information effectively without using their agent-specific communication channel.

Most research efforts on the Hanabi problem use complex architecture design with advanced algorithms, often focusing on estimating and calculating the *beliefs* of other agents [13, 26]. However, one avenue of the Hanabi problem yet to be explored is that of conventions, and how to incorporate existing human conventions into MARL solution strategies. In this paper, we propose a method to incorporate human conventions into MARL through the use of artificial *conventions*, which act as cooperative actions that span over and include multiple time steps and multiple agents, as well as show how it significantly improves on the performance of MARL agents within Hanabi for self-play [27] as well as cross-play [28] scenarios. Our approach shares similarities with options, due to the multi-time step extension of actions, however is fundamentally built on a different concept.

1.1 Related Work

A popular solution strategy to RL problems introduced by Hessel *et al.* [29], called *Rainbow*, combines various advancements made to deep Q-learning and has proven to offer significant performance gain when faced with large discrete action spaces [30–32]. A natural extension of Rainbow to multi-agent systems, is the implementation of independent Rainbow agents, referred to as multi-agent Rainbow (MA-Rainbow) [33]. However, Rainbow introduces a plethora of new hyperparameters which can often result in suboptimal policy development for cooperative settings [34], and is known for falling into suboptimal local minima [35–37].

Hanabi was first proposed as a viable medium and frontier for MARL by Bard *et al.* [37], with focus placed on the philosophical ideas and challenges found within the problem setting and how they translate to real-world scenarios. Bard *et al.* conducted initial tests for three different MARL algorithms, namely MA-Rainbow, actor-critic-Hanabi-agent (ACHA) [37], and Bayesian action decoder (BAD) [26], comparing them to state-of-the-art handcrafted bots, such as *HatBot* [38] and *WTFWThat* [39]. In the most difficult scenario of five players, the MARL solutions achieved an acceptable score of 16.8/25, however this still significantly falls behind the best handcrafted bot able to achieve a score of 24.89/25.

The simplified action decoder (SAD), introduced by Hu and Foerster [13], aims to solve a similar goal as the BAD algorithm, but with the main benefit of reducing the effect exploration actions have on the public beliefs of agents [13]. They improve on the performance of the BAD algorithm, while also extending these concepts to more than just two players. Recurrent neural networks [9] are used to track the public beliefs and an auxiliary task [18] is trained to predict if a card is playable, discardable or unknown. Hu and Foerster found that these auxiliary tasks only benefited the two-player scenario while drastically hurting the 3–5 player performance [13]. They are able to achieve good performance for 2–5 players in Hanabi, and close the gap between RL agents and handcrafted solutions by achieving a score of 22.06/25 in five-player Hanabi. However, the SAD algorithm is computationally expensive, requiring a minimum of 40 CPU cores, 2 high-end GPUs and 256 gigabytes of RAM, while still requiring billions of samples and a wall time of 72 hours [13].

Lerer *et al.* [14] further builds on the concept of the SAD algorithm by introducing multi-agent tree search (MATS) [40], and manage to achieve state-of-the-art performance for 2–5 player Hanabi. The agents manage to beat the current best handcrafted bots in the two-player scenario, with the handcrafted bots only outperforming the agents in five-player by 4%. However, just like its predecessor, the SAD-MATS algorithm is computationally expensive and requires a substantial amount of training data and wall time to achieve these results [14].

Hu *et al.* [28] also builds on the concept of the SAD algorithm, but rather than achieving state-of-the-art performance for self-play agents, they focus on the agent’s ability to cooperate with a variety of partners—a concept known as cross-play. This is achieved using *other-play*, an architecture for multi-agent systems specifically designed to break the ties formed between self-play agents, allowing them to produce more robust strategies. This allows the agents to cooperate with agents from different training runs (or regimes), and even human cooperators [28]. Even though the other-play agents are able to achieve effective cross-play performance when paired with a variety of agents in a two-player scenario, the research does not focus on higher player counts where more advanced cooperative strategies are required. Additionally, other-play builds on the existing architecture of the SAD algorithm leading to increased computational complexity and cost [28].

In contrast to value-based methods, Yu *et al.* [41] show that multi-agent proximal policy optimization (MAPPO) offers similar performance to value-based alternatives in four different cooperative tasks, namely multi-agent particle world environment (MPE) [42], StarCraft micromanagement challenge (SMAC) [43], Google research football (GRF) [44], and the Hanabi challenge [37]. MAPPO is able to match, and in some instances outperform the capabilities of leading edge value-based methods, such as Qplex [45], RODE [46], and value decomposition networks (VDNs) [10]. It shows promising results in Hanabi and often outperformed MA-Rainbow, but is sample inefficient requiring a substantial amount of training steps (more than 10 billion) to achieve good performance.

1.2 Summary of Contributions

Rather than focusing on complex architecture design, this research shifts focus to incorporating existing domain knowledge of the Hanabi problem into MARL algorithms. Hanabi has an active and dedicated community of players constantly searching for new conventions and ways to reliably beat the game. These conventions are often based on principles⁴, and have generally been standardised within the Hanabi player-base. Our main contribution is to show how these existing human conventions can be implemented in a MARL scenario using artificial *conventions*, which act as a special form of cooperative actions and can be incorporated into existing MARL algorithms using action space augmentation.

One of the key insights to our approach is the “subscribing” technique for *convention* continuation, where an agent can initiate a *convention* (based on a certain human convention) and subsequently a new action appears to the other agents to “subscribe”

⁴A convention principle is a strategy, or user defined “rule”, external to the existing game rules which governs a player’s behaviour and results in a desired outcome.

or continue with this *convention*. This strategy requires that agents opt in to a certain *convention* in order for it to reach completion, and allows an agent to halt a certain inferior *convention*, and/or initiate an improved *convention*, based on their unique observation. To this end, each *convention* contains an initial condition acting as the start of a *convention*, subsequent continuation conditions for other agents to opt in, a termination condition for a terminating action, and finally a policy that translates each step of the *convention* into an environment action based on the current agent’s observation.

Our approach shares similarities with that of multi-agent options, but addresses a problem where agents must often wait for other agents to complete their options before they can initiate a new option (which can include a cooperative option) [47]. Fundamentally, our approach is built on a different concept, due to the cooperative requirement of conventions to reach fruition, and their ability to convey ideas or intentions through implicit communication based on mutually agreed upon “rules”. Therefore, it is important to note that our approach does not introduce additional explicit communication, or additional communication channels for that matter, but rather allows for implicit communication through the nature of conventions. Furthermore, each agent forming part of a *convention* must actively choose to participate in every step of the *convention*, rather than an external policy taking control and executing over multiple time steps until the option terminates.

Ultimately, we believe our work will act as the foundation for *convention* discovery, i.e. the ability for agents to define their *conventions* as they train. Similar to how options paved the way for option discovery by showing the importance and benefits of options, *conventions* offer a similar argument for multi-agent cooperation in partially observable environments. Option discovery has been proven to be a difficult problem to solve in multi-agent systems [47, 48], and we believe *convention* discovery would offer a similar challenge. Therefore, it is important to first prove the capabilities of conventions, their implications, implementation, and limitations, especially at the hands of a difficult problem setting such as Hanabi.

Conventions result in a large performance uplift for existing MARL techniques, specifically multi-agent Rainbow, by significantly decreasing the training time and, in the case of 3–5 player Hanabi, increasing the converged performance. Additionally, since the agents are learning from a communal and well-defined list of principles, the agents are not just able to cooperate in self-play, but also in a cross-play setting with agents from different training runs (or regimes). Cross-play has been shown to be an important research area for MARL, since it opens the door to agents being able to cooperate with never-before-seen teammates, whether they are agents from other regimes, agents with different architectures, or humans (such as autonomous agents navigating the road alongside humans) [49, 50].

1.3 Paper Outline

In Section 2 we introduce the background and theory of RL, MARL, and options, in order to give context for our approach. This is followed by the introduction and discussion of *conventions* and action space augmentation with *conventions* in Section 3. In Section 4, we introduce the Hanabi problem and learning environment, as well as

existing human conventions and how they translate to *conventions* using our discussed formulations and definitions. Section 4 also contains the results for initial tests conducted on the Small Hanabi environment, to prove the capabilities and benefits of using action space augmentation with *conventions*. Finally, we present and discuss our results for self-play and cross-play for all player counts in Hanabi, in Section 5.

2 Multi-agent Reinforcement Learning and Options

Reinforcement learning aims at solving decision-making problems and is built on the concept of Markov decision processes (MDPs) [51]. Multi-agent reinforcement learning introduces more than one agent into the environment and often incorporates partial observability, and thus is built on the concept of decentralised partially observable Markov decision processes (Dec-POMDPs) [52]. The Dec-POMDP framework consists of

$$\langle S, A, T, O, P, R, n, \gamma \rangle, \quad (1)$$

where S denotes the global state space and A represents the joint-action space of the n agents at time t . The observation space O consists of the local observation of each agent i at time t within the global state S ($O_t^i = \mathcal{O}(S; i; t)$). After A is sent to the environment, the global state transitions from S to S' given the state transition probability T , and similarly P represents the observation transition probability to transition from O to O' . This results in the environment producing R , which denotes the immediate reward for each agent action that contributed to a global state change at t , with γ denoting the discount factor for future reward. These rewards are either combined with equal weighting (summed), or can have their own discount factor γ_r , and is referred to as the forward accumulated reward [52].

In value-based RL we can use equation (1) to calculate the value functions (Q) which acts as a quantitative measure for the desirability of a state. In tabular methods these values are represented using a lookup-table, while deep RL methods use powerful function approximators to estimate these values.

2.1 Independent Q-learning

Q-learning is an off-policy, temporal difference (TD) control algorithm that learns the action-value function $Q(S, A)$ by directly approximating the optimal action-value function $Q^*(S, A)$, independent of the policy being followed [51]. The update step for the action-value function is defined as

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha [R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)], \quad (2)$$

where α is the learning rate and γ the discount factor of future rewards [51]. This can be extended to MARL with independent Q-learning, where each agent has their own action-value function, conditioned on their observations O_t^i instead of the global state S_t , and using their individual experiences in their update steps [53]. The algorithm for independent Q-learning in a turn-based scenario is shown in Algorithm 1, where each agent i uses their own observations O_t^i to update their action-value functions Q^i .

The algorithm receives as input all the hyperparameters for the agent's architecture, and produces a policy as output in the form of the action-value function.

Algorithm 1 Independent Q-learning in a turn-based environment

```

1: Initialise hyperparameters: learning rate  $\alpha \in (0, 1]$ , discount factor  $\gamma \in (0, 1]$ ,  

   total number of player  $P$ , and exploration rate  $\epsilon \in (0, 1)$   

2: Initialise all  $Q(O, A) \leftarrow 0$   

3: for each episode do  

4:   Reset environment and set total time  $t \leftarrow 0$   

5:   while  $S_{t+1}$  is not terminal do  

6:     for all players  $i$  from 0 to  $P - 1$  do  

7:        $A_t^i = \text{argmax}_a Q^i(O_t^i, a)$   $\triangleright$  Choose  $A_t^i$  using  $O_t^i$  and the policy derived  

      from  $Q^i$   

8:       Take action  $A_t^i$  in the environment  

9:       Receive and store  $R_{t+1}$  and  $O_{t+1}^i$ , with  $O_{t+1}^i \in S_{t+1}$ , from environment  

10:       $\sum_{f=0}^{P-1} R_{t+1+f}$   $\triangleright$  Calculate the FAR based on the rewards for the next  

        round of actions resulting from  $A_t^i$   

11:       $Q^i(O_t^i, A_t^i) \leftarrow Q^i(O_t^i, A_t^i) + \alpha [\sum_{f=0}^{P-1} R_{t+1+f} + \gamma \max_a Q^i(O_{t+P}^i, a) -$   

         $Q^i(O_t^i, A_t^i)]$   

12:       $t \leftarrow t + 1$   

13:    end for  

14:  end while  

15: end for

```

Due to the turn-based nature of the environment, for each episode, the algorithm steps through each agent (line 6) until the global terminal state S_{t+1} is reached. Since the algorithm depicts Q-learning, the agents use an *argmax* function to select an action based on the Q-values for a given observation, shown in line 7. After an action has been chosen, it is sent to the environment, which reacts accordingly and produces a new observation and reward. Note that due to the unique nature of turn-based settings, a special type of forward accumulated reward (FAR) must be used which consists of the one round return resulting from each agent's action [41], i.e. $\sum_{f=0}^{P-1} R_{t+1+f}$. Finally, the original observation, next observation, as well as FAR are used in line 11 to update the action-value function according to equation (2).

Instead of distinct policies Q^i , independent Q-learning agents can make use of a shared policy Q , especially when the environment contains symmetries [53]. This allows the update step in equation (2) to be updated more frequently by using the experiences of all the independent agents. This has proven to offer significant performance gain and allow agents to learn more effectively [53].

2.2 Deep Q-learning

Deep Q-learning is an extension of tabular Q-learning where artificial neural networks (referred to as deep Q-networks or DQNs) are used as non-linear function approximators [54]. Deep Q-learning implements an experience replay memory to store the experiences $e = \langle S_t, A_t, R_{t+1}, S_{t+1} \rangle$ [55], which in turn is sampled in random batches b to remove correlation within the observation sequence, and thereby smoothing over the changes within the data distribution [54].

Deep Q-learning usually incorporates a *policy network* and a *target network* [54]. The policy network is used to select actions and utilises the update step, while the target network serves as a baseline when calculating the *TD-error*. The target network is updated periodically with the policy network to reduce correlation with the target. The weights of the policy network are updated using backpropagation with the goal of minimizing the TD-error, which is defined as

$$L_j(\theta_j) = \mathbb{E}_b[(R_{t+1} + \gamma \max_a Q(S_{t+1}, a; \theta_j^-) - Q(S_t, A_t; \theta_j))^2], \quad (3)$$

where θ_j and θ_j^- are the weights of the policy-and target network, respectively, at iteration j [54]. In practice the backpropagation and calculation of the TD-error is usually handled by an optimizer, such as the *Adam* optimizer [56]. Similar to tabular Q-learning, deep Q-learning can be extended to MARL using independent deep Q-learning [6], and can also make use of a shared policy.

2.3 Options

Options are built on the concept of semi-MDPs, where actions can have varying lengths over multiple time steps t as the environment transitions from S to S' [20]. When an agent chooses an option ω , the option executes stochastically according to a policy π_ω . This policy allows for existing domain knowledge to be incorporated into RL techniques. For example, in the case of a robot navigating terrain, a potential option can include: {move around boulder}, and once the agent chooses that option, an object avoidance controller takes over executing policy π_ω .

Once policy π_ω has concluded, the agent can choose a new option, effectively simplifying the problem at hand by solving it on a higher level, since the agent no longer has to focus on learning the *primitive actions*⁵ as well as the overarching problem simultaneously. Thus, an option ω is defined as

$$\langle I_\omega, \pi_\omega, \beta_\omega \rangle, \quad (4)$$

where I_ω is the initial condition that must be met for an option to become available in a certain state ($S_t \in I_\omega$), π_ω the policy according to which the option executes, and β_ω the termination condition to which the option is executed [20]. The policy π_ω can also be defined using RL, a process known as intra-option learning [20]. This has further lead to the development of *option discovery*, which uses a Laplacian framework to allow

⁵Primitive actions refer to the raw actions of the environment, for example: move up, move left, move right, move down, etc.

an agent to define and learn its own options as well as the intra-option policies while interacting with the environment [57]. Options have also been extended to MARL with Amato *et al.* [47] introducing their macro-action Dec-POMDP (MacDec-POMDP) and showing how options affect the Dec-POMDP structure.

Recently, Chen *et al.* [48] explored option discovery within MARL and introduced their own variation on the Laplacian framework using Kronecker graphs. Options within MARL poses unique challenges due to the temporal disconnect of the actions as a direct result of them having varying lengths. The actions are also chosen asynchronously, which further adds to the complexity and non-stationarity of the environment [47]. Additionally, agents in a synchronous setting must often wait for other agents to complete their options before they can initiate a new option, especially if that new option is a cooperative one [47].

3 Augmenting the Action Space with *Conventions*

Similar to how options enabled the incorporation of existing domain knowledge into RL algorithms by modifying an agent’s action space, we propose a MARL solution strategy to achieve a similar goal. Options required the defining of semi-MDPs, where actions can have varying lengths, and in a MARL scenario will theoretically require decentralised partially observable semi-MDPs (encapsulated by Amato *et al.*’s MacDec-POMDP [47]). However, our solution strategy does not directly change the length of an action by having an external policy take over, rather it facilitates implicit communication through a specific behaviour as a result of a sequence of actions, based on predefined principles, called a *convention*. Each *convention* contains *convention-steps* that span over and include multiple time steps and multiple agents, requiring each agent to actively opt in for the *convention* to reach fruition. Therefore, our augmentation of the action space does not directly change the Dec-POMDP framework, and merely acts as an extension of the existing action space to allow for more advanced behaviours.

Ideally an agent’s action space can be comprised of a pure *convention* space, i.e. containing only *convention-steps* and no primitive actions. However, there can exist scenarios where no *convention-steps* are available, and the agents will not be able to take any actions. To solve this problem, we propose augmenting the action space with primitive actions **and** *conventions* simultaneously, similar to the discussion by Sutton *et al.* [20] on combining primitive actions and options at the slight cost of performance. We will prove in Section 4.3 at the hands of the Small Hanabi problem, that this solution strategy only slightly impacts the overall performance when comparing agents with a pure *convention* space and an augmented action-*convention* space.

3.1 *Conventions*

Existing MARL approaches in partially observable environments, such as Hanabi, naturally develop conventions, but these conventions are nonsensical and vary greatly from run to run [28, 49, 50]. We propose incorporating existing human conventions into MARL algorithms using artificial *conventions*, which act as advanced cooperative actions that span over and include multiple time steps and agents. *Conventions* require

that all the agents involved participate or “subscribe” in order for it to reach fruition, however the agents cannot communicate directly which *convention* is being started or followed.

Conventions can be divided into two categories, namely available *conventions* and active *conventions*, based on if an agent can initiate the *convention* or continue the *convention*. Each *convention* contains a number of steps for that *convention* (referred to as *convention-steps*) which corresponds to the number of actions in that *convention*, allowing it to reach fruition and result in a desired behaviour. We define a *convention* c_k , containing a number of *convention-steps* m_k , as

$$c_k = \langle m_k, \lambda_k^1, \pi_k^1, \lambda_k^2, \pi_k^2, \dots, \lambda_k^{m_k}, \pi_k^{m_k} \rangle. \quad (5)$$

The set of conditions λ_k determine which *convention-steps* are currently available to an agent based on their current observation, i.e. where $O_t \in \lambda_k$ determines the available *convention-steps* $1 : m_k$. Subsequently, the set of policies π_k determine the corresponding environment action A_t for each *convention-step* with $A_t = \pi_k(O_t)$.

Thus, the initial condition λ_k^1 must be met for a *convention* to become available, and if an agent chooses to initiate c_k , the policy π_k^1 determines the corresponding environment action to start the *convention*. The *convention* will then become active, and agents can “subscribe” to c_k if they meet any of the conditions $\lambda_k^{2:m_k}$, with $\pi_k^{2:m_k}$ determining their environment action to continue c_k . Eventually an agent will have the opportunity to perform a unique continuation action to complete c_k , but only if they meet the final condition $\lambda_k^{m_k}$, with $\pi_k^{m_k}$ determining their environment action to terminate c_k and allow it to reach fruition. When implementing K *conventions* in an agent’s action space, the *convention-steps* of the different *conventions* are combined to form the *convention-step* space⁶ defined as

$$C = \langle \pi_0^1, \pi_0^2, \dots, \pi_0^{m_1}, \pi_1^0, \pi_1^1, \dots, \pi_K^{m_K} \rangle, \quad (6)$$

with

$$|C| = \sum_{j=0}^K m_j. \quad (7)$$

Conventions are further distinguished based on their step-size m_k , with single-step *conventions* having a step-size of $m_k = 1$, two-step *conventions* having $m_k = 2$, and multistep *conventions* having $m_k > 2$. Single-step *conventions*, i.e. $c_k = \langle 1, \lambda_k^1, \pi_k^1 \rangle$, contain only one condition λ_k^1 , as well as one policy π_k^1 , which simultaneously acts as the initial-and final conditions of the *convention*, allowing an agent to initiate and terminate the *convention* on a single time step. Although this might seem similar to a primitive action, single-step *conventions* still allow for additional communication through the observed behaviour resulting from the policy π_k^1 , since the policy will result in a different environment action based on the current observation, as opposed to primitive actions which remain constant independent of the current observation.

⁶Note that the number of implemented *conventions* K is not equal to the size of the *convention-step* space, since *conventions* are variable length vectors based on the number of steps m_k contained within each *convention* c_k .

Two-step *conventions* only contain two conditions and two policies, which act as the initial-and terminating steps of the *convention*, i.e. $c_k = \langle 2, \lambda_k^1, \pi_k^1, \lambda_k^2, \pi_k^2 \rangle$.

It is important to note that *conventions* do not have to be sequential, e.g. if agent 3 had their turn in between agent 1 and 2, and *convention* c_1 is active and only relevant to player 1 and 2, agent 3 would be able to initiate or subscribe to a different *convention* and c_1 will remain active. When more *conventions* are introduced, the learning problem becomes apparent and crucial, since agents must learn which *convention* is optimal in certain scenarios where more than one *convention* is available. Furthermore, since agents have the ability to opt in to a *convention*, they can also choose not to, which will result in an active *convention* terminating without a completing action. This allows agents to halt certain *conventions* based on their unique observations alluding to it being non-optimal. Thus, the agents must learn when to prioritise superior *conventions* over existing or active inferior ones. We note that the majority of our implemented *conventions*, presented and discussed in Appendix A, are single- and two-step *conventions*, with the only multistep *conventions* being the *Prompt* and the *Finesse* (each having an $m_k = 3$).

3.2 Action Space Augmentation

Conventions can either act as the only actions available to an agent, or can be incorporated into the existing action space using action space augmentation. This requires that the *convention*-step space be appended to the existing primitive-action space, and produce the augmented action-*convention* space. However, before this can be achieved, the primitive actions must be translated into unique *conventions* to ensure effective cohesion. Ultimately, a primitive action is a special type of single-step *convention* with no initial condition and a fixed deterministic policy mapping to a certain environment action independent of the current observation, i.e.

$$c'_k = \langle 1, \lambda_k^1 = \{\mathcal{O}\}, \pi_k^1 = a_k \rangle, \quad (8)$$

where \mathcal{O} represents any observation, and a the corresponding primitive action.

Thus, we can combine the primitive actions c'_k defined in equation (8) and the *conventions* c_k defined in equation (5) to produce the augmented action-*convention* space C defined as

$$\begin{aligned} C &= \{c'_0, c'_1, \dots, c'_{|A|-1}, c'_{|A|}, c_{|A|+1}, \dots, c_{|A|+K}\} \\ &= \langle \pi_0^1, \pi_1^1, \dots, \pi_{|A|-1}^1, \pi_{|A|}^1, \pi_{|A|+1}^1, \pi_{|A|+1}^2, \dots, \pi_{|A|+K}^{m_K-1}, \pi_{|A|+K}^{m_K} \rangle, \end{aligned} \quad (9)$$

with

$$|C| = \sum_{j=0}^K m_j + |A|. \quad (10)$$

This ensures that an agent has access to actions in situations where *conventions* aren't applicable or available. Additionally, this increases the learning problem complexity since an agent must learn to effectively utilise *conventions* as well as primitive actions, given scenarios where both are applicable and available.

To implement action space augmentation with *conventions*, a few changes must be made to the existing algorithm of interest. In the case of independent Q-learning shown in Algorithm 1, the algorithm will receive an additional input detailing the list of implemented *conventions*, which in turn gets appended to the existing primitive action space to produce the augmented action-*convention* space defined in equation (9). The output remains the same, i.e. a policy in the form of an action-value function, however this function will be conditioned on the augmented action-*convention* space C , rather than just the primitive action space A . Therefore, we can define Algorithm 2, which acts as the algorithm for independent Q-learning with an augmented action-*convention* space in a turn based environment.

In line 2, we define the list of *conventions* from $0 : K$, followed by the translation of the primitive actions to the special form of single-step *conventions* c'_k in line 3. The *conventions* and primitive actions are then be combined to form the augmented action-*convention* space C , shown in line 4. In line 11, the current observation, produced by the environment for the active player i at time t , determines the available and active *conventions* using the set of conditions λ_k for each *convention*. This produces the action mask to indicate which *conventions*-steps are currently available to the agent, and is often used in deep learning to ensure that an agent chooses “legal” actions given a certain state. Notably, this also encapsulates the legal primitive actions given the current observation, since they are contained within the augmented action-*convention* space. Once the action mask is determined, agent i can start or continue an available or active *convention* (or take a primitive action) by choosing a *convention*-step C_t^i based on the policy derived from Q^i , the action mask C_{mask} , and their observation O_t^i , shown in line 12.

In line 13, we use the chosen *convention*-step C_t^i to determine the corresponding *convention* k , i.e. the *convention* which contains the specific *convention*-step, and its appropriate step m , i.e. the step in that *convention* where the chosen *convention*-step occurs. The *convention* k and the step m can then be used to determine the appropriate environment action A_t^i , according to the policy π_k^m and the current observation O_t^i , shown in line 14. If the agent chose a primitive action from the augmented action-*conventions*, the step m will be equal to one and the *convention* k will correspond to the primitive action a_k , according to equation (8). Finally, the environment action A_t^i is sent to the environment which in turn produces the next observation and reward used to calculate the FAR, similar to Algorithm 1. As is the case with human players, the agents aren’t allowed to communicate which *convention* is currently being started or followed, but rather only observes the environment action determined by the appropriate *convention* policy, shown in line 15.

From the environment’s perspective, agents are merely choosing actions based on the observations the environment provided, but in reality the agents are choosing these actions based on a set of predefined *convention* policies π_k . In the case of deep RL, lines 11 and 13-14 indicate that our approach requires two new layers to be added to the agent’s architecture, namely one before the input layer to determine the available and active *conventions*, i.e. the action mask, and one at the output to translate the chosen *convention*-step C_t^i into an environment action, based on the corresponding *convention* k , the *convention*’s step m , the corresponding policy π_k^m , and the current

Algorithm 2 Independent Q-learning with an augmented action-*convention* space in a turn-based environment

```

1: Initialise hyperparameters similar to Algorithm 1
2: Define the list of conventions using  $c_k = \langle m_k, \lambda_k^1, \pi_k^1, \lambda_k^2, \pi_k^2, \dots, \lambda_k^{m_k}, \pi_k^{m_k} \rangle$ , with
    $k = 0 : K$ 
3: Translate the primitive actions to conventions using  $c'_k = \langle 1, \{\mathcal{O}\}, \pi_k^1 = a_k \rangle$ , with
    $k = 0 : |A|$ 
4: Combine the primitive actions and conventions to produce the augmented action-
   convention space  $C = \{c'_0, c'_1, \dots, c'_{|A|-1}, c'_{|A|}, c_{|A|+1}, \dots, c_{|A|+K}\}$ 
5: Initialise all  $Q(O, C) \leftarrow 0$ 
6: for each episode do
7:   Reset environment and set total time  $t \leftarrow 0$ 
8:   while  $S_{t+1}$  is not terminal do
9:     for all players  $i$  from 0 to  $P - 1$  do
10:    Obtain  $O_t^i$  from environment
11:     $C_{mask} = \{C | O_t^i \in \lambda_k\}$  ▷ Define the action mask
      using the available and active conventions determined by the set of conditions  $\lambda_k$ 
      and the current observation  $O_t^i$ 
12:     $C_t^i = \text{argmax}_c(Q^i(O_t^i, c) \cap C_{mask})$  ▷ Choose  $C_t^i$ 
      from the available and active conventions based on  $O_t^i$ , the action mask  $C_{mask}$ ,
      and the policy derived from  $Q^i$ 
13:     $m, k = C[C_t^i]$  ▷ Index the augmented action-convention space to
      determine  $m$  and  $k$ 
14:     $A_t^i = \pi_k^m(O_t^i)$  ▷ Use the convention policy to determine the
      environment action
15:    Take action  $A_t^i$  in the environment
16:    Receive and store  $R_{t+1}$  and  $O_{t+1}^i$ , with  $O_{t+1}^i \in S_{t+1}$ , from environment
17:     $\sum_{f=0}^{P-1} R_{t+1+f}$  ▷ Calculate the FAR similar to Algorithm 1
18:     $Q^i(O_t^i, C_t^i) \leftarrow Q^i(O_t^i, C_t^i) + \alpha [\sum_{f=0}^{P-1} R_{t+1+f} + \gamma \max_c Q^i(O_{t+P}^i, c) -$ 
       $Q^i(O_t^i, C_t^i)]$ 
19:     $t \leftarrow t + 1$ 
20:  end for
21: end while
22: end for

```

observation O_t . This is illustrated in Fig. 1, and in the case of deep Q-learning, the action selection step is performed using $\text{argmax}_c Q(O_t, c)$, with the available and active *conventions* as well as the primitive actions forming the action mask.

Agents with an augmented action-*convention* space can also make use of a shared policy, i.e. each agent will have access to the same network architecture depicted in Fig. 1, as discussed in Section 2.1. Since *conventions* act as an extension of the existing action space, it can be applied to any MARL method by adding the necessary *convention* components on top of the existing architecture as well as translating the

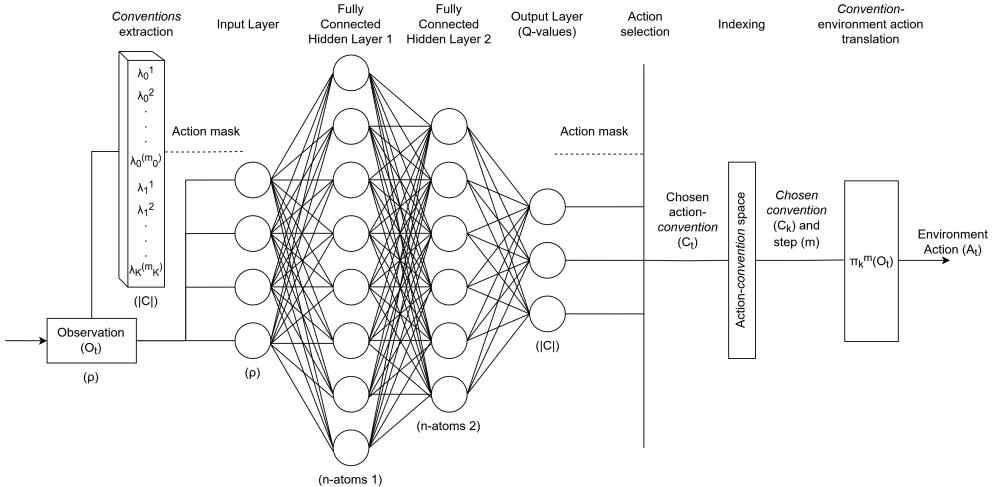


Fig. 1: Architecture design for a feed forward neural network (NN) with an augmented action-convention space applied. The environment provides the observation O_t and receives the environment action A_t . The size of the input layer is equal to the size of the observation tuple (ρ) , and each hidden layer has a size equal to the number of atoms for that layer. The size of the output layer is equal to the size of the augmented action-convention space $(|C|)$, and the action selection is determined by the MARL algorithm. Finally, the chosen augmented action-convention determines the specific convention c_k and its step m , which in turn is used to produce the environment action A_t according to the policy π_k^m and the observation O_t .

existing primitive action space into single-step *conventions*, similar to our discussion on independent Q-learning.

4 Hanabi

To test the capabilities of *conventions*, we will use the cooperative Hanabi environment, which was proposed as a frontier for MARL algorithm development by Bard *et al.* [37]. Hanabi is a 2–5 player card game, best described as a cooperative solitaire [37]. Players have a hand of five cards for two-and three-player, and four cards for four-and five-player. Each card has a suit/colour (red, yellow, green, white, or blue) and a rank (1 to 5). The deck comprises 50 cards total, with 10 cards for each suit with a distribution of three 1's, two 2's, 3's, and 4's, and finally only one 5. The aim of the game is to stack these suits in numerical order from 1 to 5, however players cannot see their own cards (i.e. their hands are hidden), but can see the cards of their fellow players.

The players take consecutive turns, and on each turn a player can either *play*, *discard*, or *hint*. *Hinting* involves revealing all the cards in another player's hand matching a certain rank *or* a certain suit, and consumes one of the limited (and shared) hint tokens. In the centre is a stack for each suit where the players must *play* their cards, and a successful play entails playing a card that follows the current card

on top of a stack (starting at 0). If the card played does not follow the current card on a stack, the play was unsuccessful (called a *misplay*) and the players lose one of their shared life tokens. *Discarding* involves removing a card from the current player’s hand and adding it to the discard pile, effectively removing it from the game, while also replenishing a hint token. An example game is shown in Fig. 2.

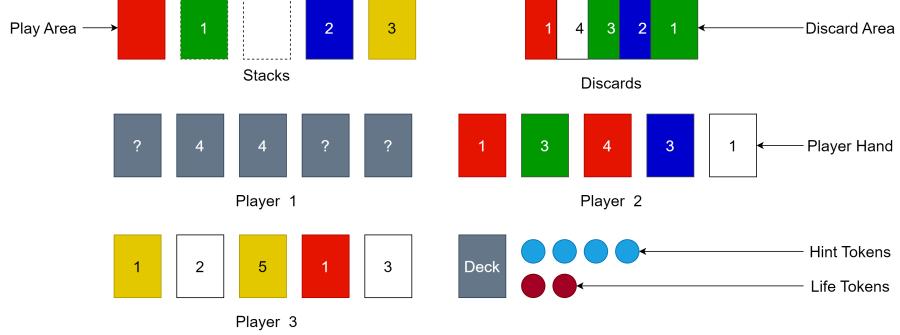


Fig. 2: An example game of Hanabi as seen from the perspective of player 1. There are four hint tokens left, and the players have lost one of their shared life tokens. Player 1 knows about two 4s in their hand and the green, blue and yellow stacks have been partially completed, leading to a current game score of 6/25. It is now player 1’s turn, and they can take the hint (to player 2 or 3), play (from their hand) or discard (from their hand) action.

At the start of the game the players have a shared total of three life tokens and eight hint tokens. If all the hint tokens are depleted, a player cannot take the hint action and must either play or discard a card from their hand. The players are awarded a shared score depending on the number of cards successfully placed on each stack. If the players manage to build the stacks to the maximum of 5 each, the game ends and the players receive a perfect score of 25/25. Alternatively, if the players lose all of their life tokens (called “bombing out”), or if the deck has been depleted, the game also ends. Note, bombing out results in a score of 0/25 independent of how many cards were played successfully, while deck depletion results in a final score equal to the current score. A discard action, or a misplay, will result in the card being removed from the game, and that player must draw a new card from the deck to replenish the missing card (which is also hidden from that player).

Due to the partial observability imposed by a player’s hand being hidden, as well as the limited communication channel in the form of hinting, Hanabi offers a very interesting and useful challenge for MARL agents to solve [37]. Actions are highly correlated and players must coordinate effectively to achieve success, often requiring advanced strategies and reasoning over the intentions of other player’s actions to reliably beat the game.

4.1 Human Conventions in Hanabi

Due to the nature of Hanabi, players have too few hint tokens to effectively convey all the needed information (colour and rank) of the 25 playable cards [37]. This becomes even more challenging as the number of players increase and hint tokens become more valuable, as well as the number of playable turns that keep decreasing⁷. Hints must therefore be used wisely, and must convey information about more than just one card. This is possible since more than one card with a similar rank or suit can be touched (or revealed) by a single hint, along with negative information⁸ being given about all the other cards in a hand. Unfortunately, this is not enough, and players who don't use additional strategies will struggle to reliably achieve a score above 15/25.

Players, therefore, require a way to convey additional information through implicit communication, while still remaining within the rules of the game. As stated by Bard *et al.* [37]: "This implicit information is not conveyed through the impact that an action has on the environment (i.e., what happens) but through the very fact that another player decided to take this action (i.e., why it happened)". Players can then reason over the actions of others, and implicitly communicate through them. This is done through the use of conventions, which are built on principles or mutually agreed upon "rules", that players develop outside the game. For Example:

Conventions Example 1. Let's assume the players in Fig. 2 have established a convention which states that if an ambiguous hint is given, the focus of that hint is on the left-most card. Therefore, using this convention, player 1 can hint to player 2 that they have two red cards in position one and three respectively (from the left), and player 2 will know (as a result of their convention) that the left-most card must be the playable red 1⁹.

Since the game's release, players have developed their own intricate and extensive conventions, often specific to their group of friends or cooperators, examples include the [Board Game Arena](#)¹⁰ and the [H-Group](#)¹¹, with the [H-Group](#) being the most widely used and adapted.

4.2 Artificial Conventions in Hanabi

To incorporate existing human conventions into MARL, we use *conventions* as described in Section 3. This process is best described at the hand of an example:

Conventions Example 2. In Fig. 2 each player is controlled by an agent, with the current player (referred to as the active player) being agent 1. The convention at hand (as mentioned in **Conventions Example 1**), which we will refer to as c_1 , states that if an ambiguous hint is given, the focus of that hint is the left-most (or newest) card. Since the condition λ_1^1 is met, i.e. player 2 has two cards in their hand that share the same colour or rank and the left-most one is playable, the action $\{\text{start } c_1\}$

⁷As the number of players increase, the cards remaining in the deck decrease, and a single player has less playable turns.

⁸Negative information refers to implied information about non-focus cards, e.g. if a player hints that two cards are green, the other cards must therefore be non-green.

⁹This is a common human convention and is based on the "Single card focus" principle as discussed by the [H-Group](#) and the implemented conventions found in Appendix A.

¹⁰<https://forum.boardgamearena.com/viewtopic.php?t=5252>

¹¹<https://hanabi.github.io/>

is available for agent 1. Agent 1 chooses to initiate c_1 which in turn gets translated by the policy π_1^1 to an environment action of {hint red to player 2}, ending player 1’s turn. It is now player 2’s turn, and they have just received an ambiguous colour hint from player 1, and given their current observation they can derive that c_1 is currently active. This triggers the subscribing (and also completing) condition λ_1^2 (the condition is completing since $m_1 = 2$), where agent 2 can now opt in to continue (and complete) c_1 . When agent 2 chooses to subscribe to c_1 , their action is translated by the policy π_1^2 to an environment action of {play card in position 1}, subsequently ending player 2’s turn and completing c_1 .

As seen by this example, and discussed in Algorithm 2, the agents are not allowed to convey directly which convention was started or followed, and merely observe the behaviour of the other player through that player’s environment action and their observation. We note that not all MARL scenarios allow for the observation of another agent’s actions, and agents will therefore only make use of their observations when determining the *conventions*. However, similar to the SMAC environment [43], Hanabi intrinsically allows the monitoring of other agents’ actions, and is included in an agent’s observation vector. Conventions Examples 1 and 2 focused on the “Single card focus” principle, as discussed in Appendix A.2, however there exists more advanced conventions that allow for improved implicit communication. These principles are all aimed at maximising the amount of information given by a single hint, in order to ensure there are enough hint tokens to effectively convey all the needed information of the 25 playable cards.

4.3 Preliminary Results on Small Hanabi

To test the viability and capabilities of *conventions*, we conduct initial tests using the Small Hanabi environment, since it offers a smaller state-action space and allows for a pure *convention* space¹². We compare a pure *convention* space and an augmented action-*convention* space, as discussed in Section 3.2, against their primitive action benchmarks. Due to Hanabi having a long list of intricate conventions, in addition to augmenting the action space with *conventions*, we experiment with the idea of simplifying the list of conventions during augmentation¹³. This simplified list consists of the conventions that were the most straightforward and natural to develop, and were also the most common ones to occur in a game. The removed conventions can be considered edge-case or specific to certain game states that are very uncommon, and often result in uncertainty regarding their effectiveness.

Small Hanabi has a deck reduced to 20 cards with only two suits, is limited to only two players, and the player hand size is reduced to two cards. Furthermore, the life tokens have been reduced to one, and the hint tokens to three, otherwise the game shares all the core mechanics as regular Hanabi. This results in the problem having a significantly smaller state-action space and allows for initial testing and development

¹²A pure *convention* space contains only *conventions*, i.e. no primitive actions, and accounts for all possible scenarios, ensuring the agents will always have a *convention* available. In Small Hanabi this is possible due to the small state-action space, but becomes a far greater challenge in the full Hanabi problem.

¹³This idea stemmed from the discussion by Sutton *et al.* [20], where they explored the capabilities of combining primitive actions with simplified options, since the primitive actions can account for the simplification at the slight cost of performance.

of algorithms, with the aim of solving the full Hanabi problem. Even though this problem is considered simpler than Hanabi, it is by no means an easy problem to solve, with a perfect score of 10/10 being difficult to achieve. Player hands are often “locked” with important cards that must not be discarded (called critical cards), hint tokens run out almost immediately, and players are often forced into no-win scenarios where they must choose a bad action or risk losing their single life token.

In these tests, we use our in-house learning environment as well as the open-sourced learning environment developed by Bard *et al.* [37]. We implement our own rudimentary conventions specific to Small Hanabi, which we developed through multiple human plays, presented and discussed in Appendix A.1. Fig. 3a shows the result for independent Deep Q-learning (DQN) with a primitive-action space compared to independent Deep Q-learning with a pure *convention* space. DQN with a pure *convention* space performs significantly better than DQN with a primitive-action space, achieving a faster training time and improved converged performance. To validate these results, and improve on the performance further, we conduct an initial test using Rainbow and the open-sourced Hanabi learning environment with the Small Hanabi preset, the results are shown in Fig. 3b. Rainbow is able to perform better than independent Deep Q-learning achieving a score of 7.6/10 compared to 5/10, and when substituting the primitive-action space for a pure *convention* space, the agents achieve significantly faster training time along with a slight convergent performance increase.

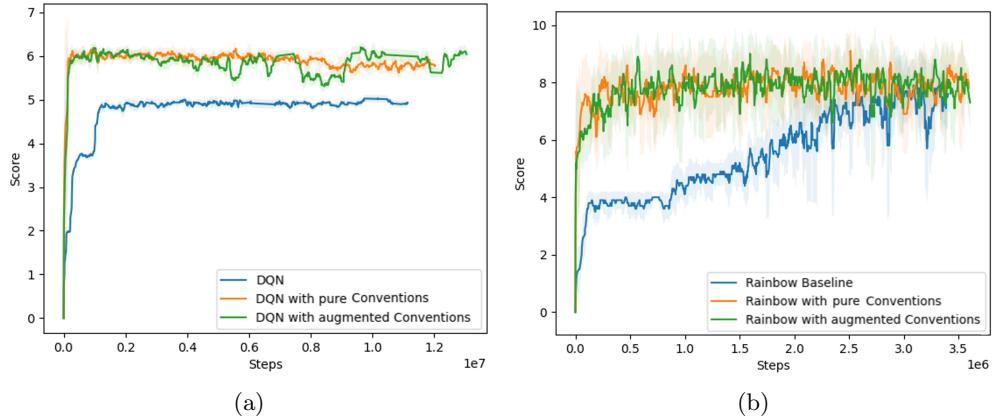


Fig. 3: (a) Learning curves for independent Deep Q-learning (DQN) with a primitive-action space, pure *conventions* space, and an augmented and simplified action-*convention* space tested in our in-house Small Hanabi environment. (b) Learning curves for Rainbow with a primitive-action space, pure *conventions* space, and an augmented and simplified action-*convention* space tested in DeepMind’s Hanabi learning environment with the Small Hanabi preset [37].

When augmenting the action space with *conventions*, we removed *conventions* 4-7 discussed in Table A1 found in Appendix A.1. Fig. 3a and Fig. 3b shows the

performance for DQN agents and Rainbow agents, respectively, with an augmented action-*conventions* space. The performance impact is almost insignificant when compared to each method’s results for a pure *convention* space. This demonstrates that even though only half of the *conventions* are present alongside primitive actions, the agents are still able to perform significantly better compared to only having primitive actions, and only slightly impacts performance compared to pure *conventions*. Therefore, when applying *conventions* in the full Hanabi problem, only a subset of established human conventions will be needed alongside primitive actions. Most existing Hanabi conventions have a list of basic conventions which are built on the most fundamental principles, and are generally considered standardised within the community. It is these conventions, as presented and discussed in Appendix A.2, alongside primitive actions that will be used when testing agents in the full Hanabi problem.

It is important to note that there could exist a combination of conventions that would result in better performance uplifts, especially when a larger variety of conventions are implemented. However for the sake of this argument, we focused on the basic conventions since they are generally the least contentious, less complex than more advanced strategies, and offer a good starting point for exploring the capabilities of action space augmentation with *conventions* in Hanabi. Further research can be conducted to explore more advance *conventions*, including advanced multistep *conventions*, and their potential performance benefits.

5 Performance Evaluation Using Hanabi

The preliminary tests for action augmentation with *conventions* in Small Hanabi has shown promising results, and lead to the agents achieving faster training times and improved policies. We now shift focus to the full Hanabi problem for 2–5 players and with the conventions defined by the H-Group and discussed in Appendix A.2. Our results will focus on Bard *et al.*’s [37] Rainbow agent as a baseline, and apply an augmented action-*convention* space to improve on its performance. Rainbow was chosen as our baseline since it is among the methods that train the fastest, is the most sample efficient among the existing Hanabi agents [13, 14, 28, 37, 41], has little run to run variance [37], is considered a widely applicable algorithm [58–60], and still leaves considerable performance to be desired within Hanabi. All tests are conducted on the open-sourced Hanabi learning environment developed by Bard *et al.* [37].

5.1 Experimental Setup

Before discussing the results, we will briefly highlight the architecture design for each deep RL technique. For a full list of the hyperparameters used in each method see Table B3 in Appendix B. All methods receive a one-hot encoded observation defined by the environment as input to their neural networks, notably (by default) this observation tuple contains the most recent actions within the previous round, and is not added as an additional feature. The Rainbow agents use a Multilayer Perceptron (MLP) consisting of two feed forward neural networks with 512 neurons each, with the *ReLU* activation function [61] applied. It implements the Adam optimizer [56] to calculate the TD-error and perform backpropagations to update the network weights.

Additionally, the Rainbow agents use distributional reinforcement learning to predict the value distributions which are approximated as a discrete distribution over 51 uniform atoms, along with prioritise replay memory sampling [29]. Even though Rainbow uses n-step bootstrapping, in these experiments a value of $n=1$ was found to be optimal, additionally noisy nets have been disabled in favour for a traditional decaying epsilon-greedy approach.

When applying *conventions* to Rainbow, we only augment the action space and leave the reward signal and observation space untouched. Additionally, we add the necessary *convention* layers to the network architecture, but keep the core algorithm unaltered, as illustrated by Fig. 1. Through testing, we found the optimal hyperparameters to be similar to baseline Rainbow, as shown in Table B3 in Appendix B. During evaluation, we compare each approach’s exponential moving averages (with a weight value of 0.9995) and standard error of the mean. All agents are trained using an *Intel Core i7 10700K CPU* and *Nvidia RTX 3080Ti GPU*. The reward type for the Hanabi learning environment is set to non-lenient, i.e. the agents will receive a large negative reward (equal to the score) when bombing out.

Due to the symmetric nature of the environment, we apply a shared policy strategy during training, i.e. the agents share an action-value function that is updated based on a communal memory of each agent’s experiences [62]. It is important to note that this still restricts learning by only using individual experiences, i.e., there is no additional sharing of state information between each agent. During the cross-play evaluation of 2–5 player Hanabi, we chose samples from 10 separately trained agents for each player count, and show the results for the combination of agents that performed the best on average.

5.2 Self-play Performance

The learning curves for Rainbow with and without an augmented action-*convention* space for 2–5 player Hanabi are shown in Fig. 4. The two-player scenario seen in Fig. 4a shows the smallest improvement when applying *conventions*, however the two-player Hanabi problem can be considered a special case. This is due to the fact that the shared information between the two players are far more limited when compared to higher player counts. For example, in two-player Hanabi, half of the hands are hidden to a single player whereas in the three-player scenario only a third of the hands are hidden, and the players always have a common hand visible. This allows for more advanced reasoning over another player’s actions, evident by the fact that two of the conventions, specifically the *Finesse* and the *Prompt* as discussed in Appendix A.2, cannot be applied in two-player Hanabi.

Despite the fact that the complexity of the problem increases exponentially as more agents are added to the environment, the performance uplift of *conventions* become more apparent. In the three-player setting seen in Fig. 4b, the agents are able to achieve a significantly faster training time as well as an improved convergent performance with a higher average score. The performance improvement becomes even more apparent when looking at the five-player setting, seen in Fig. 4d, which is generally considered the most difficult problem to solve, and where the agents train roughly five times faster than the baseline Rainbow agents. In Fig. 5 the distribution

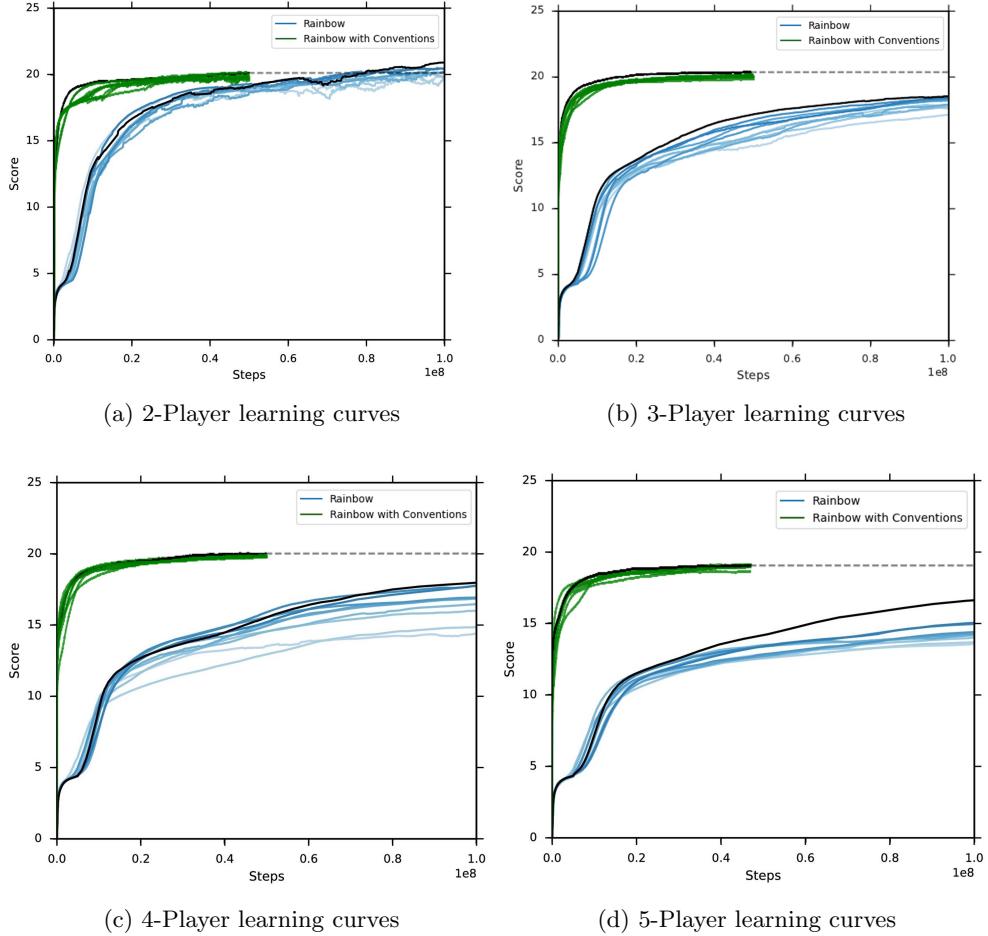


Fig. 4: Learning curves with exponential moving averages (weight=0.9995) for Rainbow, obtained from Bard *et al.* [37], as baseline compared to Rainbow with an augmented action-*convention* space for Hanabi two-to five-players. The best agent is highlighted in black for each agent scenario within each player count.

of scores during evaluation of each agent over the course of 1000 episodes is shown. In each scenario, *conventions* allow for a significant performance improvement, achieving a consistent grouping near a score of 21/25 and less overall variance when compared to baseline Rainbow. Rainbow with an augmented action-*convention* space also displays a robustness to increased player counts, with a remarkably consistent behaviour across all scenarios.

Table 1 shows the average score for the best agent from each player count’s various runs compared to the results from Bard *et al.* [37]. Rainbow with an augmented

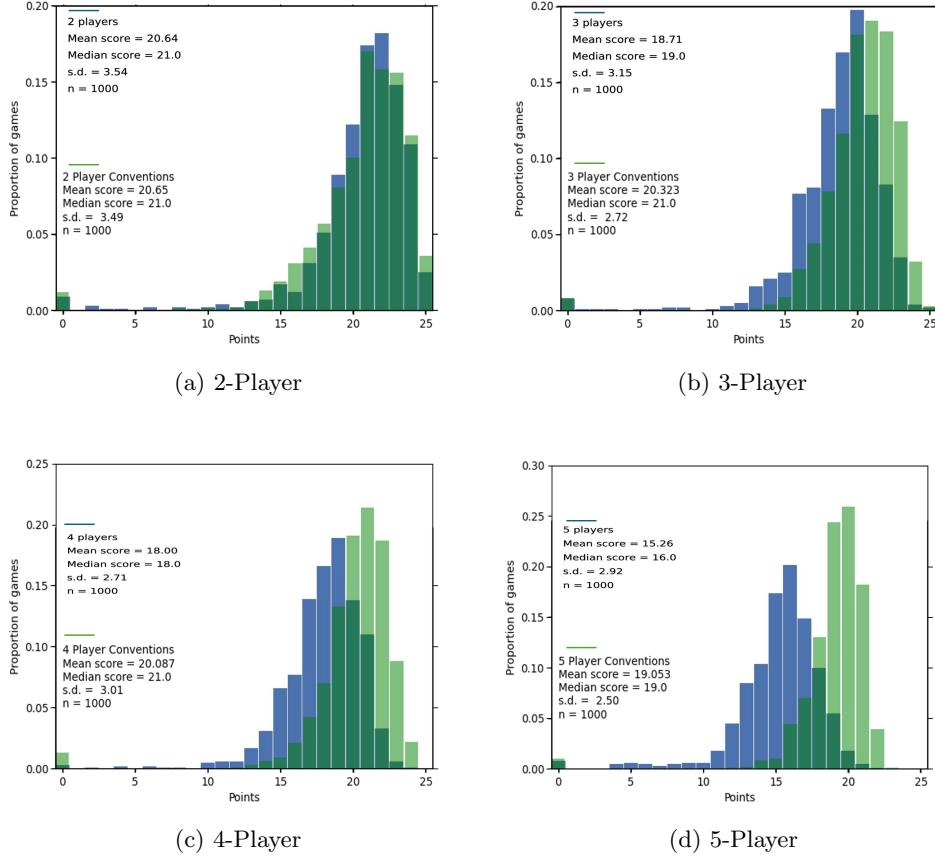


Fig. 5: Distribution of scores for 2–5 player Hanabi over the course of 1000 evaluation episodes comparing baseline Rainbow and Rainbow with an augmented action-*convention* space. The baseline Rainbow results were obtained from Bard *et al.* [37].

action-*convention* space demonstrates an improved score over baseline Rainbow for each scenario, with a higher average in the 3–5 player scenario and overall less deviation. Additionally, the three-and five-player results are competitive compared to other algorithms, such as ACHA, while also requiring substantially less training steps. In the case of five-player Hanabi, Rainbow with *conventions* outperform ACHA where ACHA required 20 billion training steps and Rainbow with *conventions* only required 30 million. A statistical significance test, conducted in Appendix C, shows that majority of the results are statistically different. However, the Rainbow with *conventions* agent’s scores are statistically similar to that of baseline Rainbow two-players and ACHA three-players, however *conventions* allow the Rainbow algorithm to reach these scores significantly faster (5x compared to baseline Rainbow and 1000x compared to ACHA). Furthermore, even though these two instances are statistically similar, the agents learn

completely different policies, as seen by the cross-play performance evaluation in the next section.

Table 1: Performance evaluation over 1000 episodes for the best performing Rainbow agent with an augmented action-*convention* space applied taken from a sample of 10 different runs for each player count, compared to the results presented by Bard *et al.* [37]. Each score is averaged out of 25 with the standard error of the mean shown in brackets, and the number of perfect games (25/25) shown as a percentage.

Method	2P	3P	4P	5P
Rainbow with Conventions	20.65 (0.11) 3.6%	20.32 (0.07) 0.2%	20.09 (0.09) 0.1%	19.05 (0.08) 0%
Rainbow [37]	20.64 (0.22) 2.5%	18.71 (0.20) 0.2%	18.00 (0.17) 0%	15.26 (0.18) 0%
ACHA [37]	22.73 (0.12) 15.1%	20.24 (0.15) 1.1%	21.57 (0.12) 2.4%	16.80 (0.13) 0%

Upon investigation, the agents tend to choose *conventions* over primitive actions 70% of the time. This is the main reason for the Rainbow agents not achieving a higher average score, since they struggle to learn some of the conventions available to them (specifically *The Chop* as discussed in Appendix A.2). We believe this to be a result of the Rainbow algorithm not having a memory of past observations, and therefore not being able to realise the importance of *conventions* that have a high payoff over extended time frames.

5.3 Cross-play Performance

Although self-play has historically led to the highest state-of-the-art agent performance within Hanabi [14], cross-play is also a crucial and widely applicable problem to solve. The ability for agents to cooperate across different training runs or regimes, or even across architectures, can greatly benefit MARL algorithms and their ability to cooperate in real-world scenarios [49, 50]. Self-play agents are notorious for not being able to cooperate with never-before-seen partners, and often results in very poor performance [28, 37, 49, 50].

Table 2 shows the two-player performance for various agents in cross-play Hanabi taken from existing literature, compared to the results of our cross-play Rainbow agents with an augmented action-*convention* space. The results for the Rainbow with *conventions* agents are statistically different from the other results reported in Table 2, as seen by Table C4 in Appendix C. *Conventions* are able to significantly improve on the performance of cross-play agents, allowing the baseline Rainbow agents to go from abysmal performance to an acceptable score. Furthermore, it is able to outperform SAD with only other-play applied for the two-player scenario, with other-play and auxiliary task applied to SAD still holding the best score. However, as mentioned previously, auxiliary tasks only benefit the two-player scenario in Hanabi [13]. When moving from self-play to cross-play, Rainbow with an augmented action-*convention* space only

suffered a slight performance decrease and maintained remarkable consistency across various agent pairings.

Table 2: Performance evaluation in two-player Hanabi for the best performing agents in a cross-play scenario over the course of 1000 episodes. The Rainbow with an augmented action-*convention* space agents were chosen from the top performing two agents in 10 different runs. Each score is averaged out of 25 with the standard error of the mean shown in brackets.

Method	2P	
	Self-play (SP)	Cross-play (CP)
Rainbow with Conventions	20.65 (0.11)	17.02 (0.25)
Rainbow [37]	20.64 (0.22)	2.91 (1.67)
ACHA [37]	22.73 (0.12)	3.31 (1.78)
SAD [28]	23.94 (0.04)	2.52 (0.34)
SAD with other-play [28]	23.93 (0.02)	15.32 (0.65)
SAD with auxiliary tasks and other-play [28]	24.06 (0.02)	22.07 (0.11)

Table 3 shows the results for cross-play Rainbow with an augmented action-*convention* space for all player counts of Hanabi. As seen in the self-play results for Rainbow with *conventions* and confirmed in Table 3, the two-player scenario is a special case, with the agents showing the biggest performance degradation when moving from self-play to cross-play, and overall worst score compared to the other cross-play results. While most research efforts on cross-play Hanabi focus on the two-player scenario, our results demonstrate that more powerful *conventions* are applicable in higher player counts, thus, it is important to consider all the player scenarios when evaluating cross-play performance.

Table 3: Performance evaluation in 2–5 players Hanabi for the best performing Rainbow with an augmented action-*convention* space agents in a cross-play scenario over the course of 1000 episodes. Each combination of agents in each player count were chosen from the top performing agents in a sample of 10 different runs. Each score is averaged out of 25 with the standard error of the mean shown in brackets.

Method	2P		3P		4P		5P	
	SP	CP	SP	CP	SP	CP	SP	CP
Rainbow with Conventions	20.65 (0.11)	17.02 (0.25)	20.32 (0.07)	18.60 (0.15)	20.09 (0.09)	18.56 (0.11)	19.05 (0.08)	17.69 (0.09)

Most remarkable is the five-player results, where the cross-play Rainbow agents with an augmented action-*convention* space are able to achieve a relatively high score and still cooperate effectively. The reason *conventions* are able to offer such a large performance gain for cross-play is due to the fact that the agents learn from a communal list of conventions that are taken from existing domain knowledge. Thus, if an agent is paired with a never-before-seen cooperator that has learned to play with the same list of conventions, they will still be able to cooperate effectively, since there

is no uncertainty over another agent’s intent and reasoning. This is not unexpected, since humans who learn from the same “list” of conventions (whether it is physical or passed down), and that have never before interacted, can cooperate and communicate effectively with relatively low uncertainty.

6 Conclusion

Hanabi offers a complex and intricate problem for multi-agent reinforcement learning agents, since it incorporates various features of real-world problems, such as those found in the autonomous agents controlling vehicles problem. It tests an agent’s ability to reason over another player’s actions and intentions, while maintaining a limited communication channel and partial observability. Existing MARL algorithms focus on complex architecture design and implement advanced algorithms capable of achieving remarkable performance. However, these algorithms are often computationally expensive, and require immense amounts of training data to learn convergent policies. In this paper, we focused on a different approach, exploring another core aspect of the Hanabi problem, namely conventions, and how to incorporate them into MARL. We showed how existing human conventions can be implemented in a MARL scenario using a special form of cooperative actions and action space augmentations.

Our main results show that *conventions* are able to significantly improve on the training time for Rainbow agents in Hanabi, and in the case of three-to five-players, also improve on the converged policy. Other Hanabi algorithms, such as SAD or MAPPO, require billions of training steps to reach convergent policies, whereas *conventions* reduced the number of training steps for Rainbow to below 30 million consistently. Additionally, *conventions* showed significant performance increase for cross-play agents, and allowed the Rainbow agents to go from not cooperating at all to being able to achieve decent performance across all player scenarios. For the most difficult scenario of five players, the cross-play Rainbow *conventions* agents trained on 50 million steps are able to outperform the ACHA self-play agents trained on 20 billion steps. Our work, as presented in this paper, mainly focused on single-and two-step *conventions* in the Hanabi problem, with the goal of extending these concepts to other multi-agent scenarios (including simultaneous action spaces) containing multistep *conventions*, as well as the exploration of more advance conventions (and the application of those conventions) on more advanced algorithms within the Hanabi setting.

An additional potential benefit of *conventions* is that of encrypted communication, where a malicious user tries to intercept the ad-hoc agent’s communications, for example military communication or team communication in sports. *Conventions* can allow for an additional means of communicating encrypted sensitive data, even if the main communication channel is compromised, since implicit communication takes place through the observed actions, and can be explored in future research. However, we believe the biggest avenue for future research is exploring and developing *convention* discovery, similar to that of option discovery, which will allow agents to produce and define their own *conventions* as they train and learn. We believe that our research will serve as the foundation for this future endeavour, proving the importance of conventions within MARL problems containing partial observability and limited communication.

Acknowledgements. This research was funded in part by the DW Ackerman Bursary as distributed by Stellenbosch University.

Author contributions. All authors contributed to the study conception and design. Material preparation and data collection were performed by F. Bredell, and all authors contributed equally to analysing the data. The first draft of the manuscript was written by F. Bredell and all authors commented on previous versions of the manuscript. All authors read and approved the final manuscript.

Declarations

- **Conflict of interest** The authors declare no competing interests.
- **Ethics approval and consent to participate** Not applicable.
- **Code availability** Code for this research is publicly available at https://github.com/FBredell/MARL_artificial_conventions_Hanabi.

Appendix A Implemented Conventions and Principles

Herein follows a list of the implemented *conventions* and the principles upon which they are built for Small Hanabi and Full Hanabi.

A.1 Small Hanabi

The principles and conventions for Small Hanabi were developed in-house through multiple human plays and, therefore, we do not claim for them to be the most optimal. However, it served as a good starting point for investigating the capabilities of *conventions*. Furthermore, these conventions allow for a pure *convention* space, i.e. the *conventions* account for all possible states and an agent will always have a convention available to take. Note that any hint convention is only available to take if there is available hint tokens to spend, as per the rules of the game.

Principles:

1. Colour saves – Players use colour hints to indicate to the other player that the card must not be discarded.
2. Value plays – Players use value hints to indicate to the other player that a card is playable, unless the value hinted card is clearly not playable (i.e. both stacks are currently above the hinted value), then the card is discardable.
3. Pseudo-chop – Players always discard oldest non-colour hinted card, unless both cards are colour hinted and a player must discard, then that player discards their oldest card.
4. Better to let go – If the current score is less than 7, a player can reveal a five in another player’s hand to indicate to that player that they must discard that five.
5. Double information – If both cards share information (i.e. colour or rank), a player can hint to another player the shared information to indicate that the newest card is playable.

6. Implied moves – If the negative information produced by other hints indicates a card as playable or discardable, a player can play or discard that card.
7. We need hints – Players should prioritise discarding for hint tokens rather than playing, especially if there is only one hint token remaining.

These principles lead to a set of *conventions*, shown in Table A1, and result in policies allowing an agent to follow their "rules". In practice each convention with its set of conditions λ_k will form part of an agent's actions in their action space. Therefore, if the agent has a pure *convention* space with the *conventions* in Table A1, the size of the *convention-step* space will be $|C| = 12$. Each *convention*, and its subsequent conditions, will become available to an agent through the use of action masking based on the current observation, as described in Section 3.2.

Table A1: Table detailing all the implemented *conventions* in Small Hanabi based on specific principles. Each *convention* c_k has a step-size of m_k , and subsequently contains a set of conditions $\lambda_k^{0:m_k}$ corresponding to the set of policies $\pi_k^{0:m_k}$ that determine the appropriate environment action A_t . However, single-step *conventions* only contain one condition and one policy, i.e. $m_k = 1$, which is simultaneously the initial and final condition and policy of that *convention*.

k	m_k	λ_k^m	π_k^m	Policy Description	Principles
0	1	λ_0^1	π_0^1	Discard oldest non-colour hinted card. If both are colour hinted, then discard the oldest card	3, 7
1	2	λ_1^1	π_1^1	Give a value hint for a playable card in the other player's hand	2
1	2	λ_1^2	π_1^2	Play a value hinted card received from the other player	2
2	2	λ_2^1	π_2^1	If the score is less than 7, and the other player has a five of any colour, value hint that five to indicate to that player that they should discard the five	4
2	2	λ_2^2	π_2^2	If the score is less than 7, and a value hinted five (which is clearly not playable) was received from the other player, discard the value hinted five	4
3	1	λ_3^1	π_3^1	If a card is critical, i.e. it must not be discarded, colour hint to the other player so save the card from being discarded	1, 3
4	2	λ_4^1	π_4^1	If a card is discardable, value hint that card to the other player to indicate that they can discard it	2, 7
4	2	λ_4^2	π_4^2	If an individual card was value hinted by the other player, and is clearly not playable, discard that card	2, 7
5	2	λ_5^1	π_5^1	If the newest card in the other player's hand is playable, and their two card share information (colour or rank), hint the shared information	5
5	2	λ_5^2	π_5^2	If the other player has hinted information about both cards in hand, the newest one is playable and can, therefore, be played	5
6	1	λ_6^1	π_6^1	If the implied knowledge reveals a card as playable, then a player can play that card	6
7	1	λ_7^1	π_7^1	If the oldest card in hand is value hinted and can be played in the future (achieved through convention 5) and hint tokens are ≤ 1 , then a player should discard their newest card (override pseudochop)	3, 5, 7

A.2 Hanabi

The principles for Hanabi are based on those defined by the [H-Group](#). We strongly encourage the reader to read their documentation, specifically chapters I – VIII, on Hanabi, conventions, and convention principles for an in-depth discussion (with examples) for each included principle and their corresponding conventions discussed below. As discussed in Section 4.3, the implemented *conventions* only consist of the most fundamental principles, since these are generally considered standardised within the Hanabi community. Majority of the listed *conventions* are single-or two-step *conventions*, however the *Prompt* and the *Finesse* are multistep. Additionally, the [H-Group](#) discuss more advance multistep conventions useful for future considerations. Note that the prompt and finesse is only applicable in three-to five-player Hanabi, however there does exist a self-prompt for two-player Hanabi which reduces the *convention* to a two-step *convention*, and is included in our implementation.

Principles:

1. The Chop – A player should discard their oldest non-hinted card, called “the chop”. If all cards are hinted and not obviously discardable, then no cards should be discarded.
2. Save Hints – A player must try to prevent another player from discarding an important card when it is on the chop by hinting the card’s colour or value.
3. Play Hints – A player must try to promise another player that a card is playable. If a hint is obviously playable (i.e. not on the chop or a value hint on the chop that follows a current stack), then it is a play hint. However, if a hint touches the chop, and it is unclear if it was a play or save hint, then assume it was a save hint.
4. Single card focus – Any hint given or received should always have one card as the focus. This principle is divided into three scenarios:
 - (a) If only one card is hinted, then that card is the focus.
 - (b) If two or more cards are hinted and one is on the chop, then the focus was on the chop (either play or save).
 - (c) If two or more cards are hinted and none are on the chop, then the focus was on the newest card (play).
5. Good touch principle – A player should only give information on cards that will be playable. A player should not give hints for cards that will never be played, ensuring that they will become “old” and become the chop (to get discarded).
6. Save principle – A player must try to save all critical cards if these cards are on the chop position. Critical cards include:
 - (a) Any five.
 - (b) A card is the last of its kind, i.e. there is no more of that card in the deck based on the discard pile.
 - (c) Any unique two, based on the current player’s perspective, i.e. none of the other players have the same coloured two in hand.
7. Minimum clue value principle – Every hint should include at least one non-hinted card.
8. Prompts – If the next two players have cards following on each other in order of play, i.e. the first player has the first card and the second player has the second card following that card, with the first player having partial knowledge of their

card (usually colour), then the current player can give a play clue to the second player. The first player will see this clue, realise it is a play clue on a card that is not yet playable, and deduce that their partially revealed card must be the card required to make the play clue valid, otherwise the second player will make a mistake.

9. Finesse – Similar to prompt, however the first player’s card is in the newest position *and* they have no knowledge of that card. This forces a blind play, since the first player will see a false play clue given to the second player, thereby realising they must have the required card to complete the play, and since none of their cards have partial information regarding the sequence (i.e. it is not a prompt), their newest card must be the required card.

There also exists implied principles, which are not directly translated into *conventions*, however they do govern some of the internal reasoning of certain *conventions*. These implied principles include:

10. Perfect teammates – A player should assume that their teammates will never make a mistake and any clue given has meaning.
11. Colour over value hints – In general, colour hints give more information than value hints and should be the preferred hint. However, there does exist situations where value hints are more beneficial and should, therefore, be used.
12. Early game – At the start of the game, if there are other *conventions* available to a player, they should not discard their chop. As soon as the first card is discarded, i.e. a player had no other option, then the chop discard becomes available to all players.

These principles and implied principles lead to the necessary set of *conventions*, shown in Table A2, and result in policies allowing an agent to follow their “rules”. Similar to the Small Hanabi *conventions*, each *convention* with its set of conditions will act as an action in the agent’s action space. If the agent has a pure *convention* space with the *conventions* in Table A2, the *convention-step* space will have a size of $|C| = 21$. However, unlike Small Hanabi, these *conventions* do not account for each possible game state, therefore, an augmented *convention-action* space is required, as discussed in Section 3.2.

As seen by the *conventions* in Table A2, the *Prompt* and the *Finesse* are the only implemented multistep *conventions* and, therefore, the only *conventions* to have a step-size $m_k > 2$. Notably, their finalising conditions λ_{10}^3 and λ_{11}^3 are identical to that of *convention* 3, i.e. λ_3^2 . The reason for this is that the environment action resulting from π_{10}^1 and/or π_{11}^1 is always a colour hint to the player after the next player (player 3 in the case of Table A2). Therefore, to complete *convention* 10 or 11, the finalising condition and policy is identical to those of *convention* 3, since from player’s 3’s perspective, they just received a clear play colour hint from the player before the previous player (for player 1 this would be player 2 as depicted in Table A2). Thus, to simplify our implementation, in practice we combined λ_{10}^3 , λ_{11}^3 , and λ_3^2 , as well as π_{10}^3 , π_{11}^3 , and π_3^2 .

Table A2: Table detailing all the implemented *conventions* in Hanabi based on discussed principles. Each *convention* c_k has a step-size of m_k , and subsequently contains a set of conditions $\lambda_k^{0:m_k}$ corresponding to a set of policies $\pi_k^{0:m_k}$ that determine the appropriate environment action A_t . For simplicity, this table shows the *conventions* applicable to player 1 in a three-player scenario, and as the player count increases or decreases, certain *conventions* also increase or decrease to account for different player counts.

k	m_k	λ_k^m	π_k^m	Policy Description	Principles
0	2	λ_0^1	π_0^1	Give a value hint for a playable card in player 2's hand (next player)	3, 4, 5, 10, 11
0	2	λ_0^2	π_0^2	Play value hinted card in hand received from player 3 (previous player)	3, 4, 5, 10, 11
1	2	λ_1^1	π_1^1	Give a value hint for a playable card in player 3's hand	3, 4, 5, 10, 11
1	2	λ_1^2	π_1^2	Play value hinted card in hand received from player 2	3, 4, 5, 10, 11
2	2	λ_2^1	π_2^1	Give a colour hint for a playable card in player 2's hand (next player)	3, 4, 5, 10, 11
2	2	λ_2^2	π_2^2	Play colour hinted card in hand received from player 3 (previous player)	3, 4, 5, 10, 11
3	2	λ_3^1	π_3^1	Give a colour hint for a playable card in player 3's hand	3, 4, 5, 10, 11
3	2	λ_3^2	π_3^2	Play colour hinted card in hand received from player 2	3, 4, 5, 8, 9, 10, 11
4	1	λ_4^1	π_4^1	If player 2's chop is a 5, then value hint to save that card from being discarded	1, 2, 6(a), 7
5	1	λ_5^1	π_5^1	If player 3's chop is a 5, then value hint to save that card from being discarded	1, 2, 6(a), 7
6	1	λ_6^1	π_6^1	If player 2's chop is a unique 2, then value hint to save that card from being discarded	1, 2, 5, 6(b), 7
7	1	λ_7^1	π_7^1	If player 3's chop is a unique 2, then value hint to save that card from being discarded	1, 2, 5, 6(b), 7
8	1	λ_8^1	π_8^1	If player 2's chop is a critical card, i.e. if that card is discarded there will not be another one for the rest of the game, colour or value hint to save that card from being discarded, based on the other cards in that player's hand and the other principles	1, 2, 5, 6(c), 7, 11
9	1	λ_9^1	π_9^1	If player 3's chop is a critical card, colour or value hint to save that card from being discarded, based on the other cards in that player's hand and the other principles	1, 2, 5, 6(c), 7, 11
10	3	λ_{10}^1	π_{10}^1	Prompt player 2 to play a partially revealed card in their hand, if player 3 has the card following that card. This is achieved by a clear play-hint to player 3, which player 2 will perceive and recognise as an "incorrect" hint, unless their partially revealed card is the one required to make it correct	3, 4, 5, 7, 8, 10, 11
10	3	λ_{10}^2	π_{10}^2	React to a prompt started by player 3, who gave a clear play-hint to player 2, however that card is not playable. Therefore, the partially revealed card in hand must be playable	3, 4, 5, 7, 8, 10, 11
10	3	λ_{10}^3	π_{10}^3	Finish prompt started by the player before the previous player, with a similar structure as <i>convention</i> $k = 3$	3, 4, 5, 8, 9, 10, 11

Table A2 Continued

k	m_k	λ_k^m	π_k^m	Policy Description	Principles
11	3	λ_{11}^1	π_{11}^1	Finesse player 2 to play their newest card in hand, if player 3 has the card following that card. This is achieved by a clear play-hint to player 3, which player 2 will perceive and recognise as an “incorrect” hint, and they have no partially revealed cards which would indicate a prompt, therefore their newest card must be the required card	3, 4, 5, 7, 9, 10, 11
11	3	λ_{11}^2	π_{11}^2	React to a finesse started by player 3, who gave a clear play-hint to player 2, however that card is not playable and there is no partially revealed card in hand that could indicate a prompt. Therefore, the newest card in hand must be playable	3, 4, 5, 7, 9, 10, 11
11	3	λ_{11}^3	π_{11}^3	Finish finesse started by the player before the previous player, with a similar structure as <i>convention k = 3</i>	3, 4, 5, 8, 9, 10, 11
12	1	λ_{12}^1	π_{12}^1	If there is a card in the chop position, then discard that card	1, 12

Appendix B Hyperparameters

Herein follows a list of the hyperparameters used for independent deep Q-learning, Rainbow, and their *conventions* variants shown in Table B3. Each method’s hyperparameters were optimized using parameter sweeps, i.e., training each method with various combinations of hyperparameters over the course of multiple runs and selecting the hyperparameters with the best results.

Table B3: Table of hyperparameter values for each method used in our evaluation of Small Hanabi and Full Hanabi 2–5 Players. The hyperparameters used for Rainbow were sourced from Bard *et al.* [37], and after testing found to be the same after implementing *conventions*.

Deep Q-learning (With and without <i>conventions</i>)		
Hyperparameter	Value	Description
Learning rate α	0.00005	Learning rate used by Adam optimizer
Discount factor γ	0.5	Discount factor used in Q-learning update step
Training exploration ϵ_t	0.001	Value of ϵ in ϵ -greedy strategy during training
Evaluation exploration ϵ_e	0.00	Value of ϵ in ϵ -greedy strategy during evaluation
Training exploration decay	1000	Decay rate of ϵ starting at 1 and ending in ϵ_t
Replay memory size	35000	Where experiences are stored to be used in the update step
Sampling batch size	64	Number of experiences randomly sampled from the experience replay memory used in the update step
Target network update frequency	200	Number of steps before the target network is updated with the policy network
Rainbow (With and without <i>conventions</i>)		
Hyperparameter	Value	Description
Learning rate α	0.000025	Learning rate used by Adam optimizer
Discount factor γ	0.99	Discount factor used in Q-learning update step
Training exploration ϵ_t	0.00	Value of ϵ in ϵ -greedy strategy during training
Evaluation exploration ϵ_e	0.00	Value of ϵ in ϵ -greedy strategy during evaluation
Training exploration decay	1000	Decay rate of ϵ starting at 1 and ending in ϵ_t
Replay memory size	50000	Where experiences are stored to be used in the update step
Sampling batch size	32	Number of experiences randomly sampled from the experience replay memory used in the update step
Target network update frequency	500	Number of steps before the target network is updated with the policy network
Distribution atoms	51	Number of distributional atoms over which the value distributions are approximated as a discrete distributions
Step count n	1	Number of steps over which the return is constructed in n-step bootstrapping

Appendix C Statistical Significance Test

Herein follows the results for a statistical significant test comparing our results for the Rainbow with *conventions* agents in self-play and cross-play against those of existing literature. For these test we use the unpaired Welch’s t-test [63], and assume a result to be statistically different from another if the calculated *p*-value is less than 0.05. We state our null hypothesis as: “The results obtained for the performance of the Rainbow with *conventions* algorithm in self-play and cross-play Hanabi is not statistically different to that of existing research on the Hanabi problem.”

These results demonstrate that majority of the Rainbow with *conventions* agents’ performances are statistically different from existing research conducted on the Hanabi problem for self-play as well as cross-play agents. The Rainbow with *conventions* agent’s scores are statistically similar to that of baseline Rainbow two-players and ACHA three-players, however *conventions* allow the Rainbow algorithm to reach these scores significantly faster (5x compared to baseline Rainbow and 1000x compared to ACHA). Furthermore, even though these two instances are statistically similar, the

Table C4: Table showing the results of a statistical significance test comparing the scores of Rainbow with *conventions* agents in self-play and cross-play Hanabi against those of existing research. If a result is statistically different, i.e. a *p*-value of less than 0.05 is obtained, the cell is coloured in green, and if a result is statistically similar it is coloured in red.

Method	2P	<i>p</i> -value	3P	<i>p</i> -value	4P	<i>p</i> -value	5P	<i>p</i> -value
Self-play								
Rainbow with Conventions	20.65 (0.11) 3.6%	–	20.32 (0.07) 0.2%	–	20.09 (0.09) 0.1%	–	19.05 (0.08) 0%	–
Rainbow [37]	20.64 (0.22) 2.5%	0.9676	18.71 (0.20) 0.2%	5.906e-14	18.00 (0.17) 0%	≈ 0	15.26 (0.18) 0%	≈ 0
ACHA [37]	22.73 (0.12) 15.1%	5.524e-36	20.24 (0.15) 1.1%	0.629	21.57 (0.12) 2.4%	2.047e-22	16.80 (0.13) 0%	≈ 0
Cross-play								
Rainbow with Conventions	17.02 (0.25)	–	18.60 (0.15)	–	18.56 (0.11)	–	17.69 (0.09)	–
Rainbow [37]	2.91 (1.67)	2.22e-16	–	–	–	–	–	–
ACHA [37]	3.31 (1.78)	5.396e-14	–	–	–	–	–	–
SAD [28]	2.52 (0.34)	≈ 0	–	–	–	–	–	–
SAD with other-play [28]	15.32 (0.65)	0.01478	–	–	–	–	–	–
SAD with auxiliary tasks and other-play [28]	22.07 (0.11)	≈ 0	–	–	–	–	–	–

agents learn completely different policies, as seen by the cross-play performance and its statistical differences.

References

- [1] Buşoniu, L., Babuška, R., De Schutter, B.: In: Srinivasan, D., Jain, L.C. (eds.) Multi-agent Reinforcement Learning: An Overview, pp. 183–221. Springer, Berlin, Heidelberg (2010). https://doi.org/10.1007/978-3-642-14435-6_7
- [2] Cao, Y., Yu, W., Ren, W., Chen, G.: An overview of recent progress in the study of distributed multi-agent coordination. IEEE Transactions on Industrial Informatics **9**(1), 427–438 (2013) <https://doi.org/10.1109/TII.2012.2219061>
- [3] Hüttenrauch, M., Šošić, A., Neumann, G.: Guided deep reinforcement learning for swarm systems. arXiv preprint arXiv:1709.06011 (2017) <https://doi.org/10.48550/arXiv.1709.06011>
- [4] Branavan, S.R., Chen, H., Zettlemoyer, L., Barzilay, R.: Reinforcement learning

- for mapping instructions to actions. In: Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP, pp. 82–90 (2009)
- [5] Busoniu, L., Babuska, R., De Schutter, B.: A comprehensive survey of multiagent reinforcement learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* **38**(2), 156–172 (2008) <https://doi.org/10.1109/TSMCC.2007.913919>
 - [6] Tampuu, A., Matiisen, T., Kodelja, D., Kuzovkin, I., Korjus, K., Aru, J., Aru, J., Vicente, R.: Multiagent cooperation and competition with deep reinforcement learning. *PLoS one* **12**(4), 0172395 (2017) <https://doi.org/10.1371/journal.pone.0172395>
 - [7] Claus, C., Boutilier, C.: The dynamics of reinforcement learning in cooperative multiagent systems. *AAAI/IAAI* **1998**(746-752), 2 (1998)
 - [8] Matignon, L., Laurent, G.J., Le Fort-Piat, N.: Independent reinforcement learners in cooperative Markov games: a survey regarding coordination problems. *The Knowledge Engineering Review* **27**(1), 1–31 (2012) <https://doi.org/10.1017/S0269888912000057>
 - [9] Hausknecht, M.J., Stone, P.: Deep recurrent Q-Learning for partially observable MDPs. In: *AAAI Fall Symposia*, vol. 45, p. 141 (2015)
 - [10] Sunehag, P., Lever, G., Gruslys, A., Czarnecki, W.M., Zambaldi, V., Jaderberg, M., Lanctot, M., Sonnerat, N., Leibo, J.Z., Tuyls, K., Graepel, T.: Value-decomposition networks for cooperative multi-agent learning. *arXiv preprint arXiv:1706.05296* (2017) <https://doi.org/10.48550/arXiv.1706.05296>
 - [11] Rashid, T., Samvelyan, M., Witt, C.S., Farquhar, G., Foerster, J., Whiteson, S.: Monotonic value function factorisation for deep multi-agent reinforcement learning. *Journal of Machine Learning Research* **21**(178), 1–51 (2020)
 - [12] Foerster, J., Assael, I.A., Freitas, N., Whiteson, S.: Learning to communicate with deep multi-agent reinforcement learning. In: *Advances in Neural Information Processing Systems*, vol. 29 (2016). https://proceedings.neurips.cc/paper_files/paper/2016/file/c7635bfd99248a2cdef8249ef7bfbef4-Paper.pdf
 - [13] Hu, H., Foerster, J.N.: Simplified action decoder for deep multi-agent reinforcement learning. In: *International Conference on Learning Representations* (2020)
 - [14] Lerer, A., Hu, H., Foerster, J., Brown, N.: Improving policies via search in cooperative partially observable games. In: *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, pp. 7187–7194 (2020). <https://doi.org/10.1609/aaai.v34i05.6208>

- [15] Ng, A.Y., Harada, D., Russell, S.: Policy invariance under reward transformations: Theory and application to reward shaping. In: Icml, vol. 99, pp. 278–287 (1999)
- [16] Randaløv, J., Alstrøm, P.: Learning to drive a bicycle using reinforcement learning and shaping. In: ICML, vol. 98, pp. 463–471 (1998)
- [17] Devlin, S., Kudenko, D.: Theoretical considerations of potential-based reward shaping for multi-agent systems. In: Tenth International Conference on Autonomous Agents and Multi-Agent Systems, pp. 225–232 (2011). ACM
- [18] Jaderberg, M., Mnih, V., Czarnecki, W.M., Schaul, T., Leibo, J.Z., Silver, D., Kavukcuoglu, K.: Reinforcement learning with unsupervised auxiliary tasks. arXiv preprint arXiv:1611.05397 (2016) <https://doi.org/10.48550/arXiv.1611.05397>
- [19] Mirowski, P., Pascanu, R., Viola, F., Soyer, H., Ballard, A.J., Banino, A., Denil, M., Goroshin, R., Sifre, L., Kavukcuoglu, K., *et al.*: Learning to navigate in complex environments. arXiv preprint arXiv:1611.03673 (2016) <https://doi.org/10.48550/arXiv.1611.03673>
- [20] Sutton, R.S., Precup, D., Singh, S.: Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. Artificial Intelligence **112**(1), 181–211 (1999) [https://doi.org/10.1016/S0004-3702\(99\)00052-1](https://doi.org/10.1016/S0004-3702(99)00052-1)
- [21] Silver, D., Huang, A., Maddison, C.J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., *et al.*: Mastering the game of Go with deep neural networks and tree search. Nature **529**(7587), 484–489 (2016)
- [22] Tesauro, G.: Temporal difference learning and TD-Gammon. Communications of the ACM **38**(3), 58–68 (1995)
- [23] Berner, C., Brockman, G., Chan, B., Cheung, V., Debiak, P., Dennison, C., Farhi, D., Fischer, Q., Hashme, S., Hesse, C., Józefowicz, R., Gray, S., Olsson, C., Pachocki, J., Petrov, M., Pinto, H.P.d.O., Raiman, J., Salimans, T., Schlatter, J., Schneider, J., Sidor, S., Sutskever, I., Tang, J., Wolski, F., Zhang, S.: Dota 2 with large scale deep reinforcement learning. arXiv preprint arXiv:1912.06680 (2019) <https://doi.org/10.48550/arXiv.1912.06680>
- [24] Premack, D., Woodruff, G.: Does the chimpanzee have a theory of mind? Behavioral and Brain Science 1, 515–526 (1978)
- [25] Lewis, D.: Convention: A Philosophical Study, 1st edn. Blackwell Publishers Ltd, Hoboken, New Jersey (2008)
- [26] Foerster, J., Song, F., Hughes, E., Burch, N., Dunning, I., Whiteson, S., Botvinick, M., Bowling, M.: Bayesian action decoder for deep multi-agent reinforcement

- learning. In: International Conference on Machine Learning, pp. 1942–1951 (2019). PMLR
- [27] Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., Lillicrap, T., Simonyan, K., Hassabis, D.: Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm. arXiv preprint arXiv:1712.01815 (2017) <https://doi.org/10.48550/arXiv.1712.01815>
 - [28] Hu, H., Lerer, A., Peysakhovich, A., Foerster, J.: "Other-Play" for zero-shot coordination. In: International Conference on Machine Learning, pp. 4399–4410 (2020). PMLR
 - [29] Hessel, M., Modayil, J., Van Hasselt, H., Schaul, T., Ostrovski, G., Dabney, W., Horgan, D., Piot, B., Azar, M., Silver, D.: Rainbow: Combining improvements in deep reinforcement learning. In: Thirty-second AAAI Conference on Artificial Intelligence (2018). <https://doi.org/10.1609/aaai.v32i1.11796>
 - [30] Harrold, D.J.B., Cao, J., Fan, Z.: Data-driven battery operation for energy arbitrage using rainbow deep reinforcement learning. Energy **238**, 121958 (2022) <https://doi.org/10.1016/j.energy.2021.121958>
 - [31] Yang, J., Yang, M., Wang, M., Du, P., Yu, Y.: A deep reinforcement learning method for managing wind farm uncertainties through energy storage system control and external reserve purchasing. International Journal of Electrical Power & Energy Systems **119**, 105928 (2020) <https://doi.org/10.1016/j.ijepes.2020.105928>
 - [32] Wang, R., Chen, Z., Xing, Q., Zhang, Z., Zhang, T.: A modified rainbow-based deep reinforcement learning method for optimal scheduling of charging station. Sustainability **14**(3), 1884 (2022) <https://doi.org/10.3390/su14031884>
 - [33] Harrold, D.J.B., Cao, J., Fan, Z.: Renewable energy integration and microgrid energy trading using multi-agent deep reinforcement learning. Applied Energy **318**, 119151 (2022) <https://doi.org/10.1016/j.apenergy.2022.119151>
 - [34] Zhao, Y., Borovikov, I., Rupert, J., Somers, C., Beirami, A.: On multi-agent learning in team sports games. arXiv preprint arXiv:1906.10124 (2019) <https://doi.org/10.48550/ARXIV.1906.10124>
 - [35] Ceron, J.S.O., Castro, P.S.: Revisiting rainbow: Promoting more insightful and inclusive deep reinforcement learning research. In: International Conference on Machine Learning, pp. 1373–1383 (2021). PMLR
 - [36] Toromanoff, M., Wirbel, E., Moutarde, F.: Is deep reinforcement learning really superhuman on atari? leveling the playing field. arXiv preprint arXiv:1908.04683 (2019) <https://doi.org/10.48550/arXiv.1908.04683>

- [37] Bard, N., Foerster, J.N., Chandar, S., Burch, N., Lanctot, M., Song, H.F., Parisotto, E., Dumoulin, V., Moitra, S., Hughes, E., Dunning, I., Mourad, S., Larochelle, H., Bellemare, M.G., Bowling, M.: The Hanabi challenge: A new frontier for AI research. *Artificial Intelligence* **280**, 103216 (2020) <https://doi.org/10.1016/j.artint.2019.103216>
- [38] Cox, C., De Silva, J., Deorsey, P., Kenter, F.H., Retter, T., Tobin, J.: How to make the perfect fireworks display: Two strategies for hanabi. *Mathematics Magazine* **88**(5), 323–336 (2015)
- [39] Wu, J.: Github - wuthefwasthat/hanabi.rs: Hanabi simulation in rust. [Online; accessed 17 July 2024]. <https://github.com/WuTheFWasThat/hanabi.rs>
- [40] Brown, N., Sandholm, T.: Superhuman ai for multiplayer poker. *Science* **365**(6456), 885–890 (2019) <https://doi.org/10.1126/science.aay2400>
- [41] Yu, C., Velu, A., Vinitsky, E., Gao, J., Wang, Y., Bayen, A., WU, Y.: The surprising effectiveness of ppo in cooperative multi-agent games. *Advances in neural information processing systems* **35**, 24611–24624 (2022)
- [42] Lowe, R., Wu, Y.I., Tamar, A., Harb, J., Pieter Abbeel, O., Mordatch, I.: Multi-agent actor-critic for mixed cooperative-competitive environments. *Advances in neural information processing systems* **30** (2017)
- [43] Vinyals, O., Babuschkin, I., Czarnecki, W.M., Mathieu, M., Dudzik, A., Chung, J., Choi, D.H., Powell, R., Ewalds, T., Georgiev, P., *et al.*: Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature* **575**(7782), 350–354 (2019)
- [44] Kurach, K., Raichuk, A., Stanczyk, P., Zajac, M., Bachem, O., Espeholt, L., Riquelme, C., Vincent, D., Michalski, M., Bousquet, O., *et al.*: Google research football: A novel reinforcement learning environment. In: *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, pp. 4501–4510 (2020). <https://doi.org/10.1609/aaai.v34i04.5878>
- [45] Wang, J., Ren, Z., Liu, T., Yu, Y., Zhang, C.: Qplex: Duplexdueling multi-agent q-learning. In: *International Conference on Learning Representation* (2021)
- [46] Wang, T., Gupta, T., Mahajan, A., Peng, B., Whiteson, S., Zhang, C.: Rode: Learning roles to decompose multi-agent tasks. In: *International Conference on Learning Representation* (2021)
- [47] Amato, C., Konidaris, G., Kaelbling, L.P., How, J.P.: Modeling and planning with macro-actions in decentralized pomdps. *Journal of Artificial Intelligence Research* **64**, 817–859 (2019) <https://doi.org/10.1613/jair.1.11418>
- [48] Chen, J., Chen, J., Lan, T., Aggarwal, V.: Scalable multi-agent covering option

discovery based on kronecker graphs. Advances in Neural Information Processing Systems **35**, 30406–30418 (2022)

- [49] Strouse, D., McKee, K., Botvinick, M., Hughes, E., Everett, R.: Collaborating with humans without human data. Advances in Neural Information Processing Systems **34**, 14502–14515 (2021)
- [50] Carroll, M., Shah, R., Ho, M.K., Griffiths, T., Seshia, S., Abbeel, P., Dragan, A.: On the utility of learning about humans for human-ai coordination. Advances in neural information processing systems **32** (2019)
- [51] Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction vol. 1, 2nd edn. The MIT Press, Cambridge, Massachusetts (2018)
- [52] Oliehoek, F.A., Amato, C.: A Concise Introduction to Decentralized POMDPs vol. 1, 1st edn. Springer, Cham, Switzerland (2016)
- [53] Tan, M.: Multi-agent reinforcement learning: Independent vs. cooperative agents. In: Proceedings of the Tenth International Conference on Machine Learning, pp. 330–337 (1993)
- [54] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G., *et al.*: Human-level control through deep reinforcement learning. Nature **518**(7540), 529–533 (2015)
- [55] Lin, L.J.: Self-improving reactive agents based on reinforcement learning, planning and teaching. Machine learning **8**(3), 293–321 (1992) <https://doi.org/10.1007/BF00992699>
- [56] Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2017) <https://doi.org/10.48550/ARXIV.1412.6980>
- [57] Machado, M.C., Bellemare, M.G., Bowling, M.: A laplacian framework for option discovery in reinforcement learning. In: International Conference on Machine Learning, pp. 2295–2304 (2017). PMLR
- [58] Schrittwieser, J., Antonoglou, I., Hubert, T., Simonyan, K., Sifre, L., Schmitt, S., Guez, A., Lockhart, E., Hassabis, D., Graepel, T., *et al.*: Mastering atari, go, chess and shogi by planning with a learned model. Nature **588**(7839), 604–609 (2020)
- [59] Luong, N.C., Hoang, D.T., Gong, S., Niyato, D., Wang, P., Liang, Y.-C., Kim, D.I.: Applications of deep reinforcement learning in communications and networking: A survey. IEEE communications surveys & tutorials **21**(4), 3133–3174 (2019) <https://doi.org/10.1109/COMST.2019.2916583>
- [60] Zhang, C., Patras, P., Haddadi, H.: Deep learning in mobile and wireless networking: A survey. IEEE Communications surveys & tutorials **21**(3), 2224–2287

- (2019) <https://doi.org/10.1109/COMST.2019.2904897>
- [61] Schmidt-Hieber, J.: Nonparametric regression using deep neural networks with ReLU activation function. *The Annals of Statistics* **48**(4), 1875–1897 (2020)
 - [62] Kraemer, L., Banerjee, B.: Multi-agent reinforcement learning as a rehearsal for decentralized planning. *Neurocomputing* **190**, 82–94 (2016) <https://doi.org/10.1016/j.neucom.2016.01.031>
 - [63] Welch, B.L.: The generalization of ‘student’s’ problem when several different population variances are involved. *Biometrika* **34**(1-2), 28–35 (1947) <https://doi.org/10.1093/biomet/34.1-2.28>