Working Paper:

Awareness and Adaptability in Multi-Agent Reinforcement Learning

Petra Vysušilová

Charles University, Faculty of Mathematics and Physics, Prague, Czech Republic

Our world converges to a multi-agent system, where interaction between different technologies and even on the Human-AI level becomes increasingly important. Reinforcement learning, as learning from experienced interaction, will hopefully meet the requirements of scalability, and no need for either labeled datasets or another form of human insights during training. This survey focuses on cooperation, adaptability, and modelling of other agents in the context of multi-agent reinforcement learning. We overview possible benchmarks and approaches. We also offer a new perspective on the popular goals and discuss their connection to abilities of awareness and adaptability.

Updated October 20, 2022

1 Introduction

Various AI solutions are more and more parts of our lives – intelligent assistants, the Internet of Things, autonomous cars, traffic control, mechanization of production and distribution, improving the use of data, or helping with research. All these AI solutions need to work together and also coexist (and better cooperate) with humans in one diverse ecosystem. In this survey, we explore two main abilities which we consider interconnected and important to acquire for agents in such human/AI environment we are converging to. These two abilities are **awareness** of others, and **adaptability**. We will also discuss related abilities and multi-agent system (MAS) tasks, especially focusing on cooperation. We are exploring these topics in the context of multi-agent reinforcement rearning (MARL) as it is currently a very promising approach, which will hopefully bring better scalability to problems originally solved by planning, and has the potential to produce adaptive and continuously learning agents. With growing computing power we want to delegate as much problem-solving time as possible to computers. The world is demanding more and more automation, requiring AI to solve more complicated, real-world problems. There is, naturally, no desire in human resources for hand-crafted rules or manually annotated datasets, and the history of deep learning, image recognition, or natural language processing constantly shows that less human expert knowledge incorporated in the learning process, better the results are (Campbell, Hoane Jr, and Hsu, 2002; Devlin et al., 2019; Sutton, Bowling, and Pilarski, 2022). Reinforcement learning, as learning from the experienced interaction, seems to be a perfect fit for this need, which causes RL to be a thriving research domain.

This text does not aim to be an exhaustive overview of all related papers and methods in the broad areas of MAS and MARL. We would rather show the variability of tasks and their solutions, the richness and problems of existing evaluation methods, and offer a different point of view from the awareness and adaptability perspective, which we would like to conduct future research on.

Text structure

The first part of the survey is dedicated to the review of basic concepts of multi-agents systems, reinforcement learning, and game theory. Feel free to skip these sections, if you are familiar with the field. We than continue in Section 3 with discussion on different benchmarks and cooperation metrics, and finally sections 4 - 6 describes awareness, adaptability, existing approaches and relationships in more details. Critical evaluation and summary together with the future work proposals can be found in conclusion.

2 Background

We will briefly present a background of multi-agent system (MAS) and reinforcement learning (RL) needed to understand other parts of the survey. Advanced readers can skip to a Section 4 and refer to this section only if needed. We will frequently refer to Hanabi, Overcooked, and Flatland MAS benchmarks through this section. If you are not familiar with these problems, they are, together with any other benchmarks, described in section 3.2.

2.1 Multi-Agent Systems

Multi-agent system (MAS) consists of two or more autonomous agents. Each agent has some form of sensors and actuators to interact with the environment and with other agents (Wooldridge, 2009). Such agents can be embodied AI programs, e.g. robots, pure software agents or even animals, or humans. Interaction between agents can be direct (e.g., communication, multi-peer actions like building alliances, interchanging, or attacking) or indirect via the influence on the environment. We can divide MAS according to several criteria (described in the following text): environment type, setting, and agent population.

Environment Properties According to (Russell et al., 1995), we can describe environments using the following properties:

- Partial or full observability:
  Fully observable environment provides all information for planning the optimal solution. Nevertheless, computing optimal solution is often infeasible due to the dimensionality which makes many problems intractable.

- Determinism/stochasticity:
  Learning in a deterministic environment is certainly easier than in the stochastic one. In addition to the environment, other agents can also be a source of stochasticity in MAS. They can simply perform exploration action once in a while or they can choose between actions

with some probability whether to achieve better performance or for being unreadable to opponents.

- Episodic/Sequential:
  Episodic environments, for example, many board games, are divided into episodes with a fresh start, reset of all environments properties, obtained rewards, etc. Meanwhile, sequential problems never come back to the starting position, and changes made earlier in the environment will influence the environment forever. Sequential environments can be finite or infinite. In the finite case, we could model the as a very long episodic task. The episodic task can also be repetitive, thus it consists of episodes with a fresh environment start, but agents can remember previous plays. For example in Iterated Prisoner's Dilemma (Press and Dyson, 2012), the environment (available actions, rewards, etc.) are independent of previous episodes, but the agent remembers if their opponent betrayed them previously or not, which could certainly affect their future decisions.

- Static/Dynamic:
  Static environment, either deterministic or stochastic, does not change its way of working (rewards, transition probabilities) and the state except after the agent's action. Other agents in the environment however also cause the dynamism of the environment, if we look at this from the single agent point of view.

- Discrete/Continuous
  We can discuss the discreteness of the formal model of environment states, inputs, and actions. Easier and much more common in the literature is an environment with discrete states and actions, although the research is shifting towards more complex problems, e.g., computer games with visual information, where the environment state space (screen frames) is continuous. Usually considered action space is discrete, which can be achieved by dividing continuous action space into discrete actions.

Settings The most important source of variance between multi-agent systems is *a setting*. The setting of such a system can be cooperative, competitive, or mixed depending on the goal each agent have (Robinson and Goforth, 2005). The goal in the **cooperative** setting is the same for all agents: to maximize the common gain. A frequent example of **competitive** setting is two-player zero-sum game (e.g. chess or go), which served well for the development of single-agent RL algorithms in the last decade. A pure competitive setting with three and more players is, however, hard to imagine. Even competitive games with one winner require typically cooperation in some game stages. The most natural type of setting is a **mixed** case when agents have reasons both to cooperate and compete (Dafoe et al., 2020). Human society is, after all, an example of such a system, where every agent has its own goals and preferences, but agents cannot be completely selfish, because they also benefit from the common welfare. We can further divide mixed settings into those where common interest is greater than conflict interest and vice versa (Leibo et al., 2021). Examples of existing benchmarks of respective types can be found in Figure 1 and in Section 3.

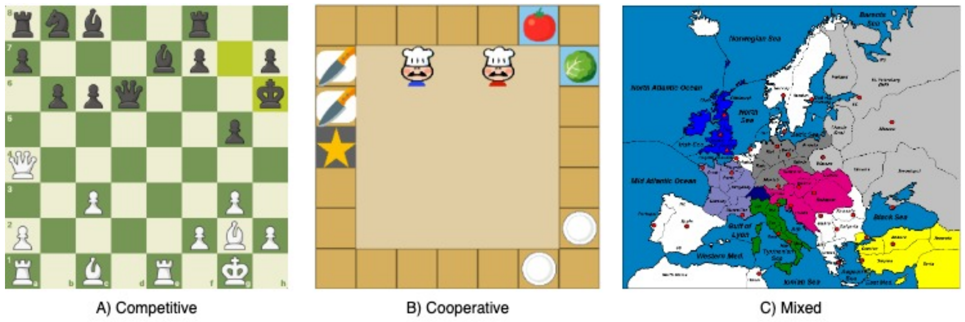

Figure 1: Examples of all three MAS settings: cooperative, competitive and mixed setting - Chess, Overcooked and Diplomacy (see section 3.2 for more info).
sources: chess.com, (Wang et al., 2020) and `http://www.amzi2.com/AINewsletter/newsletters/aix_0409.htm`

Diversity of Agents Agents can differ in many ways which, together with the environment properties and setting, brings diversity and potential difficulties in MAS problems. We can categorize them using three properties: character, abilities, and motivation. As a **character** of agent, we have in mind different agent architectures, different training processes, and different policies. Through different policies, we can also express the altruism or hostility of agents. Some agents also could be optimal and some could be not. **Motivation** of agents can differ because of different obtained rewards, therefore due to different goals and preferences, or can be influenced by an exploration effort or another introduced stochasticity. Agents can also vary in their **abilities** – they can obtain different information for example due to their field of view size, different positions, or different use of a communication channel. They can also have a diverse set of possible actions.

2.2 Cooperation abilities

The majority of problems require some kind of cooperation, therefore is a need for the following key abilities (Dafoe et al., 2020):

- **Understanding** of other agents, their goals, beliefs, and future behavior.
- **Communication** among agents for sharing information and coordinating actions.
- The ability of **commitments**, so an agent can step back from its goals in favor of the common good.
- Centralized or decentralized **institutions**, e.g., legal systems or norms which introduce more predictability of the system, prevent undesired situations, and help to solve repeatedly occurring problems, or coordinate ties breaking.

This survey focuses mainly on understanding other agents and agents adaptability, although all four abilities are interconnected and also adaptability and awareness of others it essential for all four cooperation skills.

In our opinion, understanding someone's actions (=modelling other agents) or their words (=*communication*) seems to be closely related and we would like to encourage more research on the matter of transfer learning between these two tasks. Communication is, of course, a much more complicated problem, because action space is way larger, especially in the case of arbitrary long strings from the relatively large alphabet, e.g. natural languages. Additionally, communication problem lies not only in understanding someone's message. There is a need to decide what information should be shared, how to encode it (e.g., in natural languages there are many ways how to say the same thing), and to whom it should be sent (broadcasting all information to others would explode the decision problem space for each agent) and, of course, how to use an obtained information.

The ability of *commitment* can also be related to understanding. For example, in the Flatland benchmark which represents a train network, your team-mate may need to pass through the train junction first to reach their goal, although it can mean a delay for you. Another example from the Overcooked, simulation of a restaurant kitchen, your co-working cook may need to get some tomatoes and you must step away. For all that, you need to understand your teammate intentions to help him.

Emergent norms can be very helpful in the organization of a group of AI agents or humans, but they can also be undesirable. For example in Hanabi game, norms like indicating red or yellow color if the player's newest card is playable can be very helpful if team members are not changing (Foerster et al., 2019). Both emergent and predefined types of *conventions* are useful for coordination – breaking ties in the same way, or convergence to the same equilibria. One extreme approach is a completely deterministic learning process, which will lead to the coordination of the same equilibria. It is, however, a question of whether this is still a type of cooperation we want to achieve. It definitely does not solve the problem of adaptation. An inability to adapt to new agents (e.g. new team-mates in a cooperative game) is the big problem of (not only) self-play training scheme and arbitrary norms can make it worse (Hu et al., 2020). In the Hanabi example, even the agents trained with the same algorithm only in the different experiment can have different norms, e.g., indicating blue and green in the same situation. Most algorithms and agent policies are unfortunately not able to deal with game symmetries, although some successful attempts were already made (Fickinger et al., 2021; Moravčík et al., 2017).

Multi-agent systems face lot of problems on the way to previously mentioned abilities. Many multi-agent problems are at least NP-hard. After all, even the problem of a single agent in the environment lies in PSPACE ("The Computational Complexity of Agent Design Problems" 2000) and more agents bring additional complexity. Many single-agent algorithms run into a problem with the curse of dimensionality while being applied straightforwardly to multi-agent problems (details in Section 2.4). One way of solving dimensionality problems is a *decentralization* – let every agent be an independent entity that is able to cooperate and convert the problem back to the single-agent space, although while being aware of the more complex environment. One promising approach to training such agents is the reinforcement learning (RL). When we train an agent with RL, we want him either to cooperate or to compete. As for the competitive part in MAS, basic strategies besides learning the problem solution (e.g., play the game well) is to exploit the opponent's weaknesses and not having a transparent and deterministic strategy, thus the opponent cannot do the same thing. Good performance towards the competitive surroundings is, to some extent, presented naturally in RL algorithms, meanwhile, the cooperation and the more sophisticated interaction in more-than-two-agents systems is still an open problem (Nalepka et al., 2021).

Another area of problems comes from group heterogeneity as described more in Section 4, as first MARL algorithms usually used self-play to train a group of architecturally identical agents. We will take a deeper look at possible MAS challenges in the Section 4.5. This section will continue with presenting the most important general single- and multi-agent reinforcement learning techniques, before we explore specifics of the agent modelling, adaptability, and cooperation in the rest of the survey.

2.3 Singe-Agent Reinforcement Learning (SARL)

It is obligatory to mention great success of reinforcement learning (RL) in the last twenty years in games like Chess (Campbell, Hoane Jr, and Hsu, 2002), Go (Schrittwieser et al., 2020; Silver et al., 2016), Shogi (Schrittwieser et al., 2020), Checkers (Schaeffer et al., 1992), Poker (Moravčík et al., 2017), DOTA2 (*OpenAI Five*), Starcraft (Vinyals et al., 2019), or Atari (Mnih et al., 2013). RL has some advantages over other machine learning methods, namely no need for gold data and potential for innovative/non-human solutions.

Generally, RL is good for environments where generating and running the simulation is easy and cheap as an agent is learning an optimal policy for action selection by a trial-and-error interaction with the environment.

The reinforcement learning problem is formally defined as a Markov Decision Process (described in Figure 2). We will present a definition of a partially observable variant, Partially Observable Markov Decision Process (POMDP) (Pineau, Gordon, and Thrun, 2006).

POMDP *is an 8-tuple* $(\mathcal{S}, \mathcal{A}, \Omega, R, P, O, \rho_0, \gamma)$, *where:*

- $\mathcal{S}$ *is a set of environment states,*
- $\mathcal{A}$ *is a set of actions available for the agent*
- Ω *is a set of all possible observations,*
- $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ *is a reward function defining reward in the step* t *as* $r_t = R(s_t, a_t, s_{t+1})$,
- $P(s_{t+1} = s' | s_t, a_t)$ *is a transition probability from one state to another,*
- $O : A \times \mathcal{A} \to \mathcal{P}(\Omega)$ *is a observation model where* $O(\omega_{t+1} | a_t, s_{t+1})$ *defines the probability of observing* ω_{t+1} *after action* a_t *led to the state* s_{t+1},

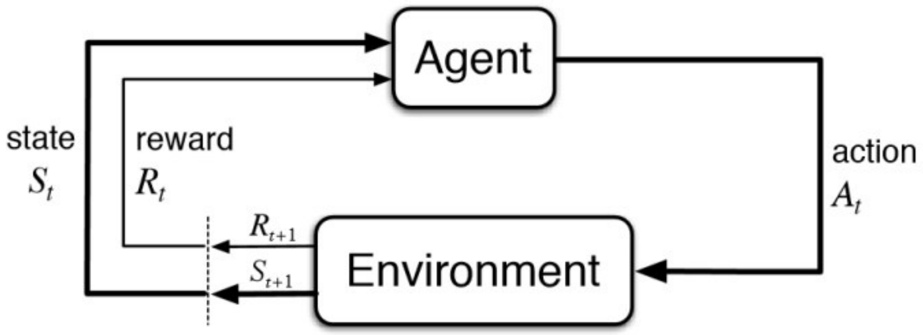

Figure 2: Reinforcement learning diagram. In every time step t, agent observes an environment state s_t, performs an action A_t, which leads to the change of the environment state from state S_t to S_{t+1} and agent also obtains a reward R_{t+1}. *source: Figure 2 of (Sutton and Barto, 2018)*

- ρ_0 *is an initial state distribution,*
- $\gamma \in [0, 1]$ *is a discount factor used for indefinite episodes.*

Markov property of the process means, that the current state of the environment depends only on the previous one, not on the whole history:

Markov Property *Process has a Markov property if for the process, where s_t is the state in the step t, if fulfils for every step:*

$$P(s_t|s_{t-1}, s_{t-2}, ..., s_{t0}) = P(s_t|s_{t-1}).$$

The RL goal is to learn an optimal *policy* $\pi : S \to A$, therefore learn what to do in every time step to maximize the expected sum of rewards:

Expected sum of rewards

$$\mathbb{E}_\pi(\sum_{k=0}^{\infty} \gamma^k R_{k+1}|a_k = \pi(s_k))$$

.

We can evaluate policy using state-value or action-value functions.

State-value function express what value assigns policy π to state s, thus how good this state is and what is the expected reward if the agent encounters this state:

State-value function

$$v_\pi(s) = \mathbb{E}_\pi[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} | S_t = s].$$

Action-value function captures what your policy thinks about the suitability of action a in the state s again with respect to the discounted sum of rewards:

Action-value function

$$q_\pi(s, a) = \mathbb{E}_\pi[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} | S_t = s, A_t = a].$$

Optimal policy is then a policy, which has an optimal value function, therefore value function returns the maximum possible value for all states. The agent updates its policy according to the obtained reward following the update rules for the corresponding algorithm.

We can divide RL techniques according to (Buşoniu, Babuška, and De Schutter, 2008) into *model-based*, *model-free*, and *model-learning*. The distinction between these methods and also between RL and planning is not always clear (Moerland et al., 2022), but we can informally describe these types as follows.

Model-free Model-free methods do not need a model to learn how to react in different situations, therefore estimate policy or value function directly (Sutton and Barto, 2018). The most important methods are those based on temporal difference (TD)[1]. As foundation TD algorithms we can name Q-learning (Kröse, 1995), or SARSA (Rummery and Niranjan, 1994), being an inspiration for all more complicated approaches. Q-learning update changes the action-value function for the given state and action by the difference between the current value and approximated new value based on the newly obtained reward:

$$q(S_t, A_t) \leftarrow \underbrace{q(S_t, A_t)}_{\text{old value}} + \alpha[\underbrace{R_{t+1} + \gamma \max_a q(S_{t+1}, a)}_{\text{aproxim. of the new value}} - q(S_t, A_t)].$$

Sarsa uses almost the same update, except that approximation of the new value for an update in the time t is done using the q-value of action and

[1]The spectrum of TD methods is wide and can also include model-based techniques (see `https://bair.berkeley.edu/blog/2018/04/26/tdm/`.

state from the next state rather than assuming the next action to be maximizing the reward:

$$q(S_t, A_t) \leftarrow \underbrace{q(S_t, A_t)}_{\text{old value}} + \alpha[\underbrace{R_{t+1} + \gamma q(S_{t+1}, A_{t+1})}_{\text{aproxim. of the new value}} - q(S_t, A_t)].$$

The main difference between these two algorithms is that q-learning is *off-policy* update meaning that its estimator does not use the current behavior policy for the estimation, it uses the greedy policy instead.

Currently most successful algorithms use deep neural networks with an actor-critic architecture (Witten, 1977), namely DQN (Mnih et al., 2015), Rainbow (Hessel et al., 2017), DDPG (Lillicrap et al., 2019), TD3 (Fujimoto, Hoof, and Meger, 2018), and A3C (Mnih et al., 2016).

Model-based Model-based RL (Sutton and Barto, 2018) needs an access to the full dynamics of the environment. This is possible in e.g. board games, but it is very rare for real-world problems because environment dynamics are either unknown or too complicated to simulate. These techniques, however, often guarantee convergence and are quicker, because knowing the model expectedly helps with learning (Kaiser et al., 2020). Model-based methods use dynamic programming and includes methods like policy iteration (Puterman and Shin, 1978), value iteration (Bellman, 1966), or Monte Carlo Tree Search (Coulom, 2006).

Model-learning Model-learning methods are sometimes described as part of model-based methods and aim to explicitly learn the model of the environment, e.g., reconstruct the full observation, or the state of the environment, and then use classical model-based methods as if the model was known (Schrittwieser et al., 2020). While learning the model, we can learn three different things: full observations (if the environment is only partially observable) (Beck et al., 2019; Gemici et al., 2017), value function (Farquhar et al., 2018; Silver et al., 2017; Tamar et al., 2016), or the next environment state and reward (Ljung, 2017). Also (Buşoniu, Babuška, and De Schutter, 2008) defines model-based learning as applying planning or model-based methods to a learned model, but you we also combine model-based and model-free methods in a more complicated system as in AlphaZero (Silver et al., 2018). A more detailed survey on model-based methods can be found in (Moerland et al., 2022).

2.4 Multi-Agent Reinforcement Learning (MARL)

This section describes extensions of single-agent RL to multi-agent system (MAS). We can describe the multi-agent problem formally as a *Multi-Agent Partially Observable Markov Decision Process (MPOMDP)* (Boutilier, 1996). In MPOMDP all agents have the same reward, thus a major disadvantage is its limitation to a pure cooperative setting only. This straightforward extension to MAS creates a new action space by taking every possible combination of individual actions as one of the new actions, thus another problem is the exponential growth of action space which causes the algorithm to struggle to converge in a meaningful time. A more general concept is called *Stochastic Game (SG)*, which can also capture the pure cooperative or competitive settings by placing some restrictions upon rewards.

Partially Observable Stochastic Game (POSG) *(Hansen, Bernstein, and Zilberstein, 2004) is a 9-tuple of* ($\mathcal{S}$, N, $\{\mathcal{A}^i\}$, $\{\Omega^i\}$, $\{R^i\}$, $\{O^i\}$,P, ρ_0, γ) *for* $i \in 1, .., N$, *where:*

- $\mathcal{S}$ *is a set of environment states,*
- N *is a number of agents,*
- $\mathcal{A}^i$ *is a set of actions for agent* i,
- Ω^i *is a set of possible observations for agent* i,
- $R^i : S \times \mathcal{A}^\pi \times S \to \mathbb{R}$ *is a reward function for agent* i *determining the reward after action* a_t, *which changed the environment state from* s_t *to* s_{t+1}. *By* $\mathcal{A}^\pi$ *we denote the Cartesian product* $\Pi_{i \in N}\mathcal{A}^i$
- $O^i : S \times \mathcal{A}^i \to P(\Omega^i)$ *is the observation model expressing the probability of observation* $\omega^i_{t+1} \in \Omega^i$ *after action* a^i_t *led to the new state* s_{t+1},
- P *is a transition model,*
- ρ_0 *is a distribution of the initial state, and*
- γ *is a discount factor.*

Note: (PO)SG and MPOMDP both assume sequential action execution. Parallel execution is modelled by Agent-Environment Cycle Game (Terry et al., 2020b).

Besides dimensionality, another problem arising in MAS is a *non-stationarity*. We can pretend that other agents are just a part of an environment and apply SARL techniques, but other agents are learning too and this violates the

Markov property, on which many convergence proofs rely. A deep survey on this topic can be found in (Hernandez-Leal et al., 2019).

Agents are mostly trained using a typical machine learning process: agent's interactions are divided into training and execution(=testing) time. The agent is always learning during training time, and then in the execution time should be able to work without further learning, although such learning would be beneficial for adaptation (see 5) and maybe will be more encouraged in the future. Agents are typically trained using *self-play*: the whole group of agents is trained together and all agents are of the same type, therefore playing with clones of themselves. This can lead to good cooperation within the trained group but causes problems when unknown agents encounter the system.

If we want to apply RL in MAS, we have basically three possibilities (Gupta, Egorov, and Kochenderfer, 2017): centralized, concurrent or parameters sharing scheme (see Fig. 3).

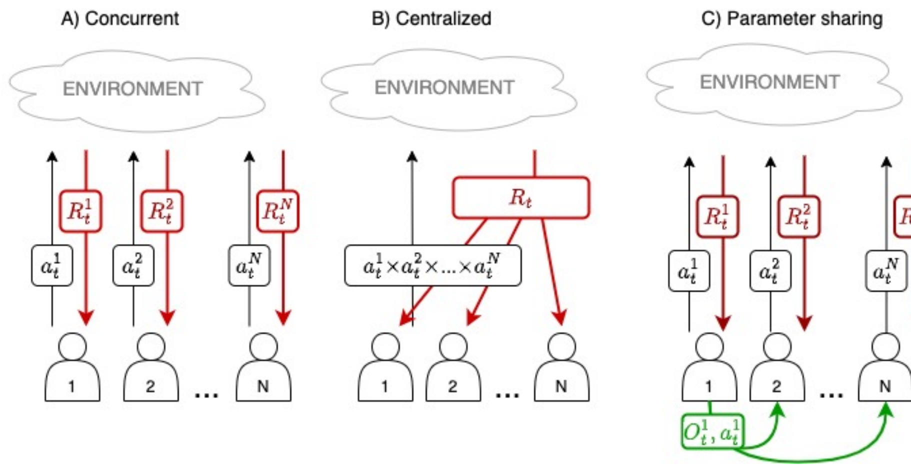

Figure 3: Training time of three main training schemes used in MARL: centralized, concurrent, and parameter sharing scheme. In the concurrent scheme, every agent has its own action to select and obtains individual reward, other agents are viewed as a part of the environment. Centralized training works with one joint action in each step and all agents obtain the same reward. In the parameter sharing scheme, agents have access to some extra information during training, for example, other agents' observations and actions. Execution time is identical to the concurrent approach.

Centralized approach Centralized approach uses joint action and trains the policy together for all present agents (Wong et al., 2021). The centralized approach has a big problem with dimensionality and therefore does not scale well, is not prepared for the incoming of new agents to the team, and requires the full observability or learning of a model of the whole environment with others. Some working attempts were however made based on the Q-learning, namely minimax-Q (Littman, 1994), hyper-Q (Tesauro, 2003), Nash-Q (Hu and Wellman, 2003), or Q-RTS (Matta et al., 2019). Although they theoretically converge only under restricted conditions, experiments showed that they can perform well even if constraints are violated from time to time.

Concurrent learning Concurrent learning means that every agent is training independently from others. This faces the problem of non-stationarity and also takes a long time because the agent cannot learn from the experience of other agents. The typical representatives of concurrent learning are variants of Winf-or-Learn-Fast (WoLF) (Bowling and Veloso, 2001) and Regret Minimization (Blum and Mansour, 2007) algorithms.

Parameter sharing scheme Parameter sharing scheme is currently the most used approach, specifically *centralized training – decentralized execution* paradigm (Kraemer and Banerjee, 2016). Agents share experience and policy parameters during training, which leads to quicker learning. Each agent has, however, different observations, and the execution is decentralized which rules out the curse of dimensionality. This approach is, unfortunately, not suitable for heterogeneous groups of agents, as all agents share the same policy. The most promising algorithms are COMA (Foerster et al., 2017), or Multi-Agent Deep Deterministic Policy Gradient (MADDPG) (Lowe et al., 2017).

2.5 Game Theory

The last research area to be presented is Game Theory, which connects AI and economy and provides a background for studying the theoretical properties of algorithms and games themselves. We will offer informal definitions of the most important concepts of game theory here.

2.5.1 Classical Game Theory

The most basic problem formulation in game theory is a normal-form game (Mas-Colell, Whinston, and Green, 1995) (see Table 1):

Player 2 / Player 1	Defect	Cooperate
Defect	1,1	3,0
Cooperate	0,3	2,2

Table 1: This table describes rewards for both players and all possible combinations in the Prisoners' Dilemma game. When both players betray each other, they will serve 2 years in prison, but betraying if the other player remains silent pays off.

Normal-form Game *can be described using a $n \times m$ matrix, where n and m is number of possible actions for player 1 or 2 respectively. The game has only one round where both players perform their actions simultaneously and the matrix contains payoff (=reward) for each player and a combination of actions.*

Problem solved in MARL are closer to another game formalization: **extensive form** (Mas-Colell, Whinston, and Green, 1995), used for description of games with alternating moves (see Fig. 4), is, on the contrary, often used in MARL, namely in solving two-player zero-sum competitive games (e.g., chess, go, checkers).

Zero-sum *game is a game where the sum of both players' payoff is zero at the end of the game, for example, one player wins with a payoff of 1 and another loose with a payoff of -1.*

First problem of this model, which is presented also as in the majority of RL algorithms, is an assumption of *rational players*.

A rational player *(Russell et al., 1995) can be described as aiming to always perform an optimal action.*

An assumption of rational players is, however, not always true and useful in MAS, mainly for two interconnected reasons. The first reason is that real-world problems involve a very diverse set of agents including humans, and most of them are not optimal. Secondary, we usually demand some level of cooperation, where ignoring team-mate's sub-optimality can be very harmful (Carroll et al., 2019) (see 5). s

If we decide anyway to use this formalism, we can describe the desired solution in terms of a *Nash equilibrium*, or a *best response*.

Nash equilibrium *(Osborne and Rubinstein, 1994) Players are playing Nash Equilibrium joint strategy if no player can gain greater reward by changing its*

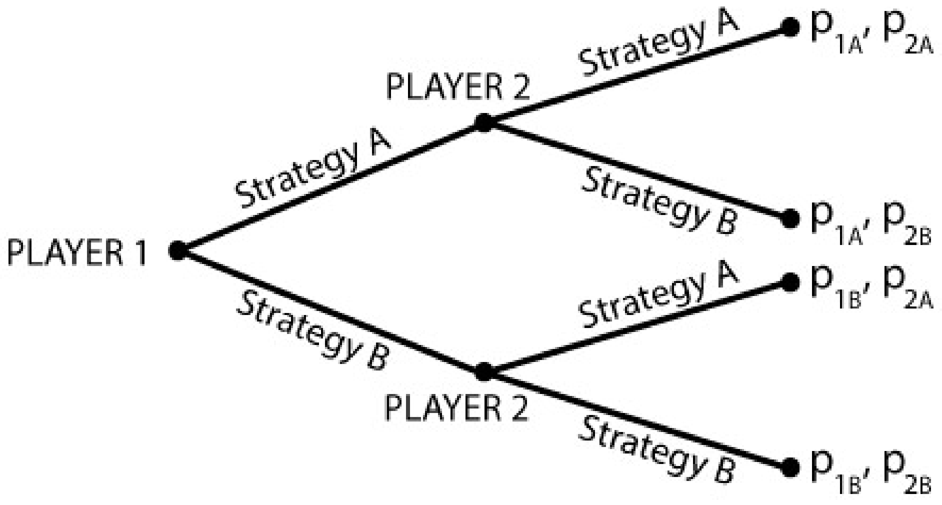

Figure 4: A general example of an extensive form game. *source: https://policonomics.com/lp-game-theory1-extensive-form*

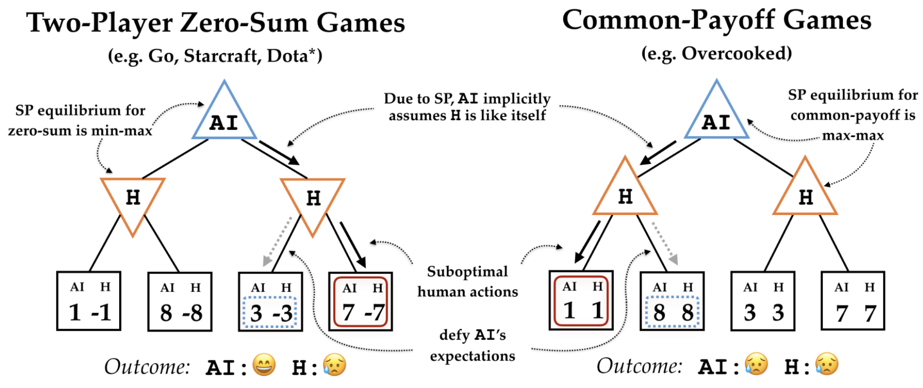

Figure 5: An illustration of suboptimal agent problem in MAS. In the competitive game (left), a suboptimal opponent will improve the reward, while in the cooperative setting (right) will cause a worse reward than expected. *source: Figure 1 of (Carroll et al., 2019)*

strategy. Beware that games can have multiple Nash equilibria with different overall payoffs.

Best response *Best response (Fudenberg and Tirole, 1991) is the policy returning the best results (reward) if other agent policies are given.*

Best response policy can act poorly against other policies than it is responding to.

2.5.2 Evolutionary Game Theory

There is very little work on the theoretical study of MARL algorithms as the classic game theory does not provide sufficient power. Recently, such theoretical properties were studied in the context of evolutionary game theory (Bloembergen et al., 2015; Tuyls, Hoen, and Vanschoenwinkel, 2006; Tuyls and Parsons, 2007), first (Börgers and Sarin, 1997), but a big unexplored space still remains. evolutionary game theory (EGT) allowed to prove convergence of q-learning in MAS in some specific occasions (e.g., two players, zero-sum game) (Kianercy and Galstyan, 2012; Tuyls, Hoen, and Vanschoenwinkel, 2006), and can be useful in the analysis of strategies to which the different algorithms converge, prediction of their behavior and therefore for tuning hyper-parameters (Bloembergen et al., 2015).

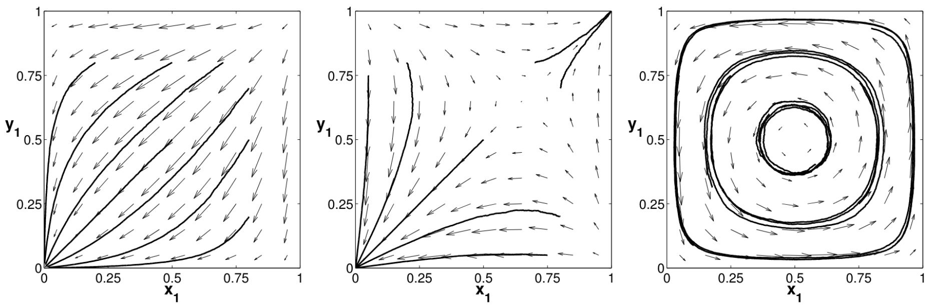

Figure 6: A visualisation of many different RL algorithms solving Matching Pennies , Prisonners' Dilema (Poundstone, 1992) and Stag Hunt (Rousseau, 1984) games. Axes represent the chosen action for player x or y respectively and arrows show the direction of convergence for both players. Precisely, we can see replicator dynamics represented with arrows, and policy traces shown by lines. These can be easily plotted for two-player two-action games, because we can fully describe the policy by the probability of action 1 of each player, denoted as x_1 and y_1, and then calculate a direction and velocity of change in each point of the matrix (=for each combination of x_1 and y_1). *source: Figure 5 of (Bloembergen et al., 2015)*

Evolutionary game theory (EGT) (Tuyls, Hoen, and Vanschoenwinkel, 2006) share the same mechanism with evolutionary algorithms. There are mutation and crossover operations applied to the population of individuals according to the *replicator dynamics*(= crossover and mutation types and probabilities, rules for crossover selection and choice of new generation, etc.). Qualities of each individual with respect to the desired result are measured by a fitness function and the whole system evolves in a way that favors individuals with greater fitness who will form a majority in the population. For this theory to be connected to playing games, we can imagine each individual as a representation of possible action. The percentage of individuals for the same action in the population represents a probability of selecting this action and the whole system evolves as the agent plays and learns. For more details, see the great survey in (Bloembergen et al., 2015).

As we have hopefully mentioned all the important concepts, we can go through the evaluation of the performance of solutions, motivation, and specific algorithms related to adaptability and awareness in the next sections.

3 Evaluation

We have an abstract goal of the adaptive cooperative agent in mind. We need, however, specific tasks for training and evaluation. This chapter will present an overview of available metrics, benchmarks, and their properties.

3.1 How to measure

The first question is how to exactly measure the performance of agents and training algorithms. In the single-agent problems, the most natural metric is averaged achieved score at the end of the game for episodic games or a score for a certain time if otherwise. Another performance measure is optimality checking – whether the agent (training algorithm) can find an optimal solution or at least potentially converge to it. This is also applicable in MAS, in pure competitive two-player zero-sum games. The situation in general multi-agent systems is far more complicated. As for optimality, the first problem is the existence of multiple optima, e.g., multiple Nash equilibria. For success, every agent needs to converge to the *same* optimum. The second problem is the presence of non-optimal agents. Such a thing is not unrealistic, humans are for example typically sub-optimal agents. In this case, optimal agents will probably achieve worse results paired with non-optimal ones (Carroll et al., 2019) and the goal is good compatibility between all agents within the system rather than optimality of the agents, because, in the realistic scenario, we will probably not train, own, or control every agent in the system. That is why this survey places such an emphasis on the adaptability of agents. If the environment will be real, thus diverse and complicated, it is more useful to have agents which are able to cooperate with humans or other non-optimal agents, even if the result will not be as good as with the team of optimal agents.

Many benchmarks want to encourage cooperation. The achieved score is not always the best description of how agents cooperate. If the environment is not forcing cooperation as the only way of success, which can be difficult to see in advance, some agents may converge to degenerated strategies. We can also want agents to specifically work great with humans, so our metric should reflect this requirement.

Some qualitative measure attempts can be found in (Nalepka et al., 2021), (Fontaine et al., 2021) and (Carroll et al., 2019),or (Baker et al., 2019) using three main approaches:

- use environment, which **forces cooperation**,

- introduce some cooperation **metrics**,
- creating **transfer** intelligence **tests**.

The problem of evaluating human-friendly agents is tackled in (Carroll et al., 2019) by setting the environment which is not doable without cooperation. Results are then evaluated against the human model and also against humans. While the first approach is limited by the quality of such a human model, experiments with humans can be long, expensive, and typically do not use a diversified group of people. Cooperation can be measured using metrics originally applied in sociology or physics (Nalepka et al., 2021), namely using recurrence quantification analysis (RQA), joint recurrence plot (JRPs) and %Determinism (%DET).

Recurrence plot *Creation of recurrence plot is a part of recurrence quantification analysis (Marwan et al., 2007). The recurrence plot represents the state's re-occurrences of a dynamical system. All positive values out of the main diagonal mean re-occurrence of state.*

JRP *Joint recurrence plot visualize the interaction between such systems, e.g., between team members. The plot visualizes the combination of repeated states for two agents (= recurrent points).*

%DET *(Brandt, 2020) measures cooperation flexibility (lower the %DET, greater the flexibility). %DET is the percentage of recurrent points from the JRPs.*

Experiments showed that more flexible agents cooperate better and such teams achieve better results.

Such techniques were first applied to AI agents in (Kim and Nam, 2020) and used in a re-evalution (Nalepka et al., 2021) of original Caroll's study (Carroll et al., 2019) with new human involving experiments. They measure not only interaction but also feedback from human users.

Simple technique of cooperation measurement is ***concurrent motion*** (Fontaine et al., 2021), which equals to overall time spend when both agents are doing something (no one is waiting). Concurrent motion detects effective cooperative work division when no agent is blocked and both can use their potential. This metric is appropriate only for a specific type of environment. There is also a need to not allow *padding actions*[2] or filter them out during calculation. Another strategy we want to reveal during testing is a *lazy*

[2]*Padding action* to have no direct impact on the result, e.g., random walking, oscillating between two tiles, etc.

agent (Sunehag et al., 2018), a situation where one agent does not do anything useful at all, which still leads to good performance, as other agents are able to achieve the goal without him.

Different approach to testing is taken in (Baker et al., 2019). They created a set of five intelligence tests as they called them – five tasks in a similar environment, with similar actions, etc but different from the original Hide and Seek game used for training. This approach does not primarily test cooperation, it tries to qualify which specific abilities agents gained during training. Specific tested abilities included manipulation with objects or object's position memory, but this method has the potential to be used for a better understanding of what agents actually learn. Also as we want some generality in agents' abilities, testing transfer possibilities of algorithms could be important.

Any evaluation technique raises questions:

- **did agents really cooperate?** (=does the environment force cooperation or at least is the metric robust against it?)
- **will this metric detect degenerated strategies**, e.g., lazy agents?
- **is it the way of cooperation we need?** Will this work with humans? Is it the most effective way how to solve the problem? Wouldn't it be better to have a different work division? etc.,

These questions should be always carefully answered before the selection of a metric. The majority of papers focus only on measuring the overall or mean per-capita score (Leibo et al., 2021)), which shifts the problem to the right choice of environment, which will not allow undesirable and degenerated strategies to be successful. More about the influence on the environment, their automatic generation, problems, and specifics will be described in the next section.

3.2 Environments and Benchmarks

First, we will discuss the automatic generation of new benchmarks and some general environments for manual benchmark creation. After that, we will provide a comparison of existing benchmarks for multi-agent systems.

Creating Benchmarks

Generating new environments is difficult and it requires a lot of human work. Prediction or analysis of such environment properties which will affect the training process in the desired way or creating environment design that would lead to new ways of behavior is very hard (Fontaine et al., 2021). We will use slightly modified terminology from MeltingPot benchmark (Leibo et al., 2021):

- **game** Game is here in the meaning of board or computer game, benchmark idea with rules, goals and a story, e.g., chess, Mario, soccer, grabbing bricks, trains on the railway network.
- **substrate** We will define substrate in a different way than (Leibo et al., 2021). Here substrate represents the specific instance of the game environment, e.g., a specific game variant with a set of rules, environment dynamics, specific map with walls, items, and other non-agent entities. Formally substrate is a partially observable Markov game.
- **scenario** Scenario is a a substrate + possible background population of agents. Background population, as defined in (Leibo et al., 2021) are pre-trained agents serving for benchmarking. Their score does not count into the result score and their presence helps diversity and adjustability of environments.
- **play** One play is one trial for specific agents in a specific scenario, whose result is measurable (typically win, loss, or some sort of numeric score). It is the same as an episode in the episodic game, or a time-bounded part of the game otherwise.

Generating new environments is difficult and it requires a lot of human work. The problem of generating scenarios automatically is dated back to 1998 (Hofer, Ramirez, and Smith, 1998) and it is an important area of research for game development. It would be nice to explore all possibilities of scenarios, each with a significant number of plays, but this is beyond computational possibilities, so we need to parametrize the environment and try only a few most promising variants. Two direct ways of environment parametrization are map generation and background population selection. We will now describe these two existing attempts at manual and automatic scenario generation.

Latest work show (Fontaine et al., 2021) that environment can substantially influence the training process. A diverse set of environments with different challenging features is essential for good evaluation, comparison of algorithms, revealing of their flaws, and understanding of how they work, which can lead to better AI algorithms in the future (Leibo et al., 2021). There is still little work on the topic of automatic environment generation, mainly focused on generating interesting maps. We may have following requirements on environment generation (Fontaine et al., 2021):

- easy to generate,
- similar to human-created environments,
- diverse, and
- with a solvability guarantee.

Proposed solutions (Fontaine and Nikolaidis, 2021; Fontaine et al., 2021), are focusing on map generation and are based on a quality diversity algorithm. They use overcooked game and by using a quality diversity algorithm they are able to generate diverse environments which minimize the team score, so they are perfect for algorithm flaws detection. Another approach, for once benefiting from the non-stationarity arising from other agent's actions, is using a fixed set of maps (human-created) and encoding diversity using other agents in the environment (= background population) (MeltingPot) (Leibo et al., 2021). MeltingPot contains a set of preprogrammed substrates and background populations, which can be extended and combined, including the creation of new maps. It is also agnostic to used training algorithms, so its use is very wide. MeltingPot predefined substrates measure well the robustness of the population and encourage the universality and generality of agents. Sometimes specialized agents and work divisions could be desirable, but this is discouraged in this benchmark. A direct way to improve testing environments diversity is by merging these two approaches, i.e. generating maps and using background populations. MeltingPot does not provide tools for automatic map generation but allows integration of maps generated externally.

We would like the creation of more general learning methods and more universal and robust agents. For this to be done, there is a need to better evaluation, which will reveal algorithm weaknesses. Quite a frequent approach to MARL algorithms evaluation is experimenting with only very few (one or two) not very general benchmarks on a limited set of scenarios. According to (Resnick et al., 2018), a successful and widely used environment should also be intuitive and fun for humans, easy to integrate, and not be too difficult for the current state of knowledge. We believe, that many benchmarks mentioned later in this section meet these requirements, and the main need is deeper research, and more extensive testing of existing and newly published algorithms on available benchmarks. Easy generation of new scenarios and especially good widely used set of diverse tasks

Table 2: An overview of MAS benchmarks

Game	**Setting**	**# Players**	**Observability**	**Input**
Hanabi	C	2-5	P	I
Overcooked	C	$2-\infty$	P	I
Diplomacy	M/P	many	P	I
Hide and Seek	P/T	$2-6$	P	V
Starcraft	P/T	1x1 - 4x4	P	V/ I
Poker	P	$2-\infty$	P	I
Flatland	C/M	$1-\infty$	F/P	I
Watch and Help	C	2	A	F
Capture the flag	P/M	∞	PF	V
ViZDoom	P	1-16	PF	V
Pomerman	P/M/T	4	P	I

Setting options: **C**= cooperative, **P**=competitive, **M**=mixed, **T**=teams. *Observability* can be **P**(=partial), **F**(=full),or **PF**(=partial first-person). *Representation* of inputs can be **I**(=internal, e.g., feature layers, signals with explicit game description etc..) or **V**(=visual, thus pixels). It is hard to categorize games because they present different types of challenges and tension points between selfish and common interests, please refer to the description of benchmarks or their source papers for better understanding. MeltingPot benchmark is in a separate table.

(like Gym (Brockman et al., 2016) and DeepMind Suite (Tassa et al., 2018) for single agent RL or MNIST (Deng, 2012) for Image Recognition) would significantly help to easily compare and select the best algorithms, which will hopefully lead to progress in Human-AI interactive and cooperative agents. Some benchmarks with this attempt already exist, namely MeltingPot (Leibo et al., 2021), Cogment (Redefined et al., 2021), or Arena (Song et al., 2020), which will be described now.

Table 3: Games included in MeltingPot benchmark

Game	Setting	# Players	Observab.	Input
Batch or Stravinsky	C/M	∞	P	I
Stag Hunt	C/M	∞	P	I
Allelopathic Hunt	C/M	∞	P	I
Collaborative Cooking: Impassable	C	2	F	I
Collaborative Cooking: Passable	C	2	F	I
Chemistry: Branched	C	∞		I
Chemistry: Cycles		∞		I
Pure Coordination in Matrix	C	∞	F	I
Rationalizable Coordination in Matrix	C	∞	F	I
Cleanup	M	∞	P	I
Prisoners' Dilemma in Matrix	P/M	2	P	I
Chickens in Matrix	M	8	P	I
Running with Scissors	P/M	2	P	I
Arena RSM	P	8	P	I
Commons Harvest: Open	P/M	∞	P	I
Commons Harvest: Closed	P/M	∞	F	I
Capture the Flag	P/T	∞	P	I
King of the Hill	P/T	∞	P	I
Territory:Open	P	∞	P	I
Commons Harvest: Partnership	C/M	∞	F	I
Territory: Rooms	C/M	∞	P	I

Setting options: **C**= cooperative, **P**=competitive, **M**=mixed, **T**=teams. *Observability* can be **P**(=partial), **F**(=full),or **PF**(=partial first-person). *Representation* of inputs can be **I**(=internal, e.g., feature layers, signals with explicit game description etc..) or **V**(=visual, thus pixels). This does not aim to be a full categorization of games, for better insight see (Leibo et al., 2021)

Exisiting Benchmarks

History Before Arena, the first benchmark designed especially for multi-agent RL, some single-agent benchmarks were naturally extended to multi-agent problems: MuJoCo (Todorov, Erez, and Tassa, 2012), ViZDoom (Wydmuch, Kempka, and Jaśkowski, 2019) and Starcraft II (Vinyals et al., 2017). MuJoCo is not a benchmark, it is rather a framework for physical modelling, which was used to create physics-based benchmarks like soccer. ViZDoom is interesting for being a visual game, therefore the source of information is an array of pixels, not some internal representation of the game state and Starcraft II is a team game.

The description of the following benchmarks is accompanied by tables 2 and 3 with a comparison of their most important properties.

General and Extensible Frameworks **Cogment** (Redefined et al., 2021) is a framework for MARL training, with special emphasis on AI-human interaction in sequential decision-making tasks. **Arena** (Song et al., 2020) and **MeltingPot** (Leibo et al., 2021) both contains pre-defined substrates or scenarios and are extensible. Arena and MeltingPot are very similar, but MeltingPot has, in our opinion, several advantages – it is newer, with a modern interface (which is quite subjective, though) and additional parametrization via background population, which leads to focus on more general agents, because allows easy training with different groups and therefore does not rely on self-play conventions. Arena, in the contrast, offers a human-AI interface and is prepared also for explicit communication, but does not allow some types of reinforcement learning, e.g., model-based RL (Chen et al., 2021). Both Arena and MeltingPot integrate many popular existing benchmarks (overcooked in MeltingPot or Capture the Flag in Arena) as part of their predefined set. We will now describe in more detail the most popular and promising benchmarks, including MeltingPot predefined substrates.

Hanabi Hanabi (Bard et al., 2020) is a card game for 2-5 players who should cooperate to win the game. The game contains cards in 5 colors and with numbers 1 to 5. Every card has one color and one number. The goal is to complete sorted sequences (starting with 1) for each color. The most interesting feature of the game is that agent does not see its own cards (just cards of other team-mates), so agents must use one of a very restricted set of allowed communication actions to indicate unknown cards to each other (see Figure 7). In each turn, the player can play one of his cards or help another player by indicating either all his cards of the same color or of the same number. If the played card can't be attached to the stack of its color, i.e., the top card of that stack is not a preceding number, one *storm token* of three is discarded[3], so a team can make only three mistakes in playing a card. Exhaustion of storm tokens leads to the end of the game. Some types of cards are presented multiple times and the number of possible help depends on the correctly played cards. At the beginning, team has 8 *note tokens* to spend on hints, and it is possible to regain spent tokens by playing a playable card. The game is NP-complete even with all cards known to everybody (Baffier et al., 2017). The game typically leads to the emergence of norms (for both humans and AI), which makes this environment interesting primarily for the exploration of ad-hoc games, i.e., games with unknown teammates, which require the creation of new norms. Another problem is the existence of many quite bad local minima (Bard et al., 2020), which complicates a convergence of self-play learning. However, some algorithms achieved a very good score in different settings: If conventions are used, a perfect score in the majority of plays can be achieved (Bouzy, 2017), and self-play SOTA results of more than 24 points out of 25 pursued in (Foerster et al., 2019), and very good HRI result can be found in (Hu et al., 2020).

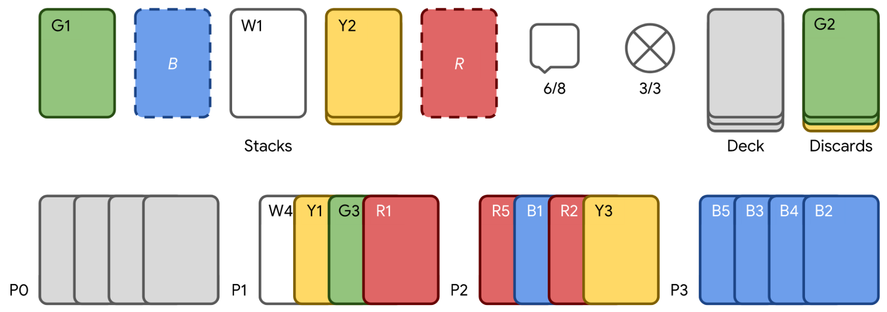

Figure 7: Hanabi game example from the player 1 point of view. No one played playable blue or red card yet. Player 0 does not know his cards and two chances to indicate someone's colors or numbers are spent. *source: Figure 1 of (Bard et al., 2020)*

[3]For full rules, see: `https://www.ultraboardgames.com/hanabi/game-rules.php`

Overcooked Overcooked (Carroll et al., 2019) is a pure cooperative game. The goal for two cooks is to serve as many dishes as possible for a fixed time or just finish one dish (it varies in different papers). In the most complicated forms, overcooked contains hierarchical tasks of cooking recipes, which are decomposable to many steps down to the simple physical-like actions like one-step move, grabbing something from an adjacent tile, etc. Different kitchen maps can force different types of behavior, e.g., force cooperation, or cause more frequent collisions to be avoided (see Fig. 8). To solve this task, agents should have three types of abilities (Wu et al., 2021).:

- working in parallel if possible,
- working together if necessary e.g., one agent can't reach some ingredients,
- collision avoidance and non-blocking movement in the space.

Many studies of Overcooked focus on the good collaboration with humans (Fontaine et al., 2021; Strouse et al., 2021; Wang et al., 2020), achieving better Human-Robot results using human-like models or by producing more adaptive agents.

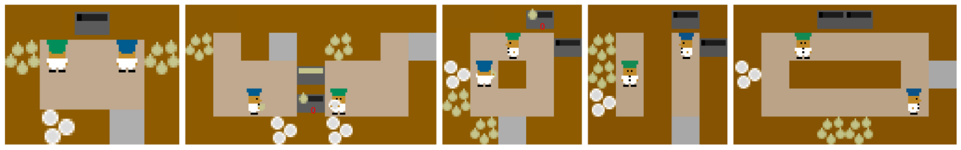

Figure 8: This figure presents some possible maps in the Overcooked game. Some of them induce a higher probability of collisions, and anothers are not doable without cooperation. *source: Figure 3 of (Carroll et al., 2019)*

Poker Poker is a card game, where players have asymmetrical imperfect information. Poker has many variants differing on the number of players, bet restrictions, number of rounds, cards, whether the cards are dealt publicly, etc. AI solutions are typically offered for easier mainly two players poker variants like heads-up limit Texas hold'em (Bowling et al., 2015) or Heads-up no-limit Texas hold'em (Moravčík et al., 2017). These variants are games for two players. Each player has two secret cards, hidden from each other. There are three rounds of dealing publicly with other cards and after each revealing of cards, each player can bet some amount of money. The tricky part is that every player has different partial information about the state of the game, and no one knows, what will be next in the deck. Even in this simplified version of Poker, there is about 10^{160} decision points (Johanson, 2013) (see Fig. 9). Games like this are very hard to solve because of the recursive nature of reasoning. One player's action can be without knowledge of his cards interpreted in many ways: *Was he bluffing? What cards can he have, that bring him too much confidence for a big bet? Why this public card changed his behavior?* The opponent is doing the same type of reasoning and some of his actions can be affected by what he believes about our action, which is influenced by what we believe about him, etc.

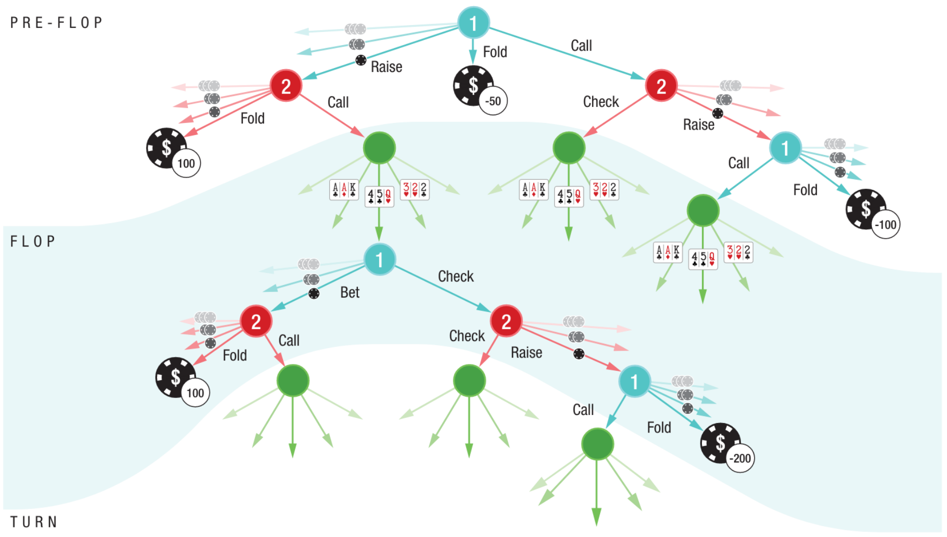

Figure 9: The part of the Poker game tree. *source: Figure 1 of (Moravčík et al., 2017)*

Diplomacy Diplomacy (Bakhtin et al., 2021) is a board game for 7 players, although restricted version for two players – France vs Austria (Bakhtin et al., 2021) – also exists. The game represents the war over 34 provinces on the map of Europe (see Fig. 10. The goal of the game is to conquer most of the territories. One player can't achieve this alone, so at least in some game stages, there is a need for cooperation and alliance forming. Another difficulty of the game comes from simultaneous actions in every step. The game tree is significantly larger than for example Chess or Go: 1021 to 1064 legal joint actions in every step, so searching the game tree for optimal action is impossible. In the real board game, players are allowed to communicate, so they can form alliances or negotiate with their enemies. Mostly used AI variant is *no-press* diplomacy, so the agents can't use communication at all. It is a simplification in the terms of action selection, but it is harder to find out what other players plan to do. Currently best results are in (Bakhtin et al., 2021). Diplomacy, being a competitive game without teams, arises a question of evaluation – should we measure the performance of all agents being trained by our algorithm or one out agent against different strategies? And if so, what should be the other strategies?

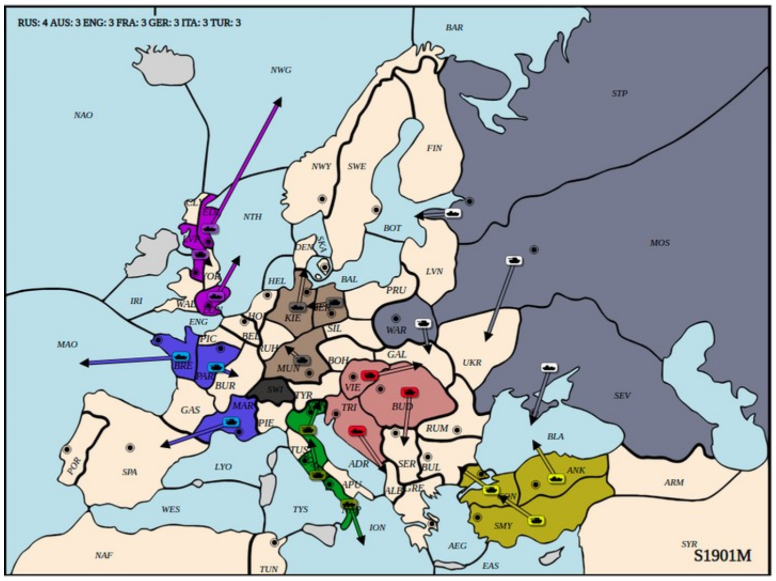

Figure 10: The illustration of the game of Diplomacy. *source: (Paquette et al., 2019)*

Flatland Flatland (Mohanty et al., 2020) is a multi-agent benchmark inspired by trains. There can be from one to many agents(=trains). The common goal is to get all trains from start to finish as quickly as possible. The environment has also a competitive part: every train wants to achieve its goal as quickly as possible, which can cause blocks or collisions for other trains. The optimal solution is not the greedy strategy for all trains. The environment can be used for classical planning algorithms as well as for reinforcement learning. It is possible to define custom vision area and information which will an agent obtain. Figure 11 illustrates the environment. In-

8

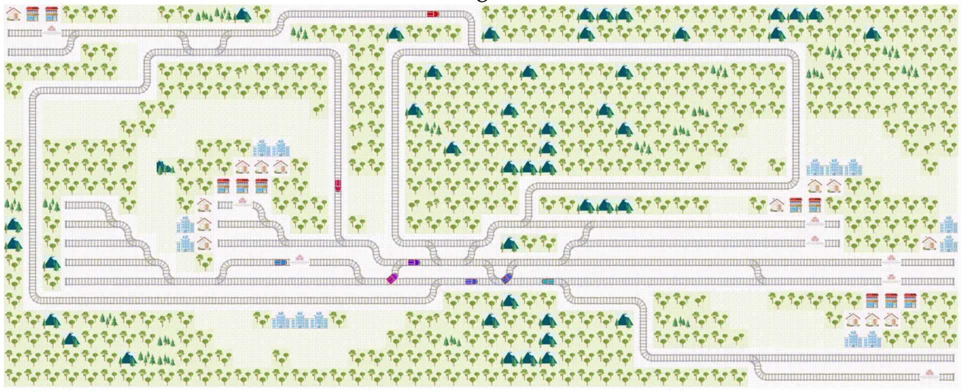

Figure 11: The example of the flatland map with trains. *source: https://flatland.aicrowd.com/intro.html*

teresting solutions were presented in the 2021 Flatland Challenge (Laurent et al., 2021) and we can also mention works on Multi-Agent Path Finding problems (Damani et al., 2021; Sartoretti et al., 2019) which will be hopefully applicable in the future.

Starcraft II Starcraft II is an environment based on a multi-player computer game with a partially observable map. Information is full visual RGB pixels or is represented by feature layers (pictures representing the most important parts of the original frame). There are about 300 possible actions and an agent needs to follow the sequence of meaningful progress: gaining resources, using them to build production buildings, producing weapons and building an army, and finally destroying all opponent's buildings, which is the ultimate goal of the game.

Watch and Help Watch and Help (Puig et al., 2021) is an environment for one AI agent, but the goal is to help the second player achieve the task, so it can be considered a multi-agent system. This benchmark has two stages. In the first stage, the agent observes the teammate's behavior and tries to figure out their goal. In the second stage, the agent tries to help with the task (see example in Figure 12).

Hide and Seek Hide and Seek (Baker et al., 2019) is a visual, physics-based game for two teams. One team's goal is to hide from the oppo-

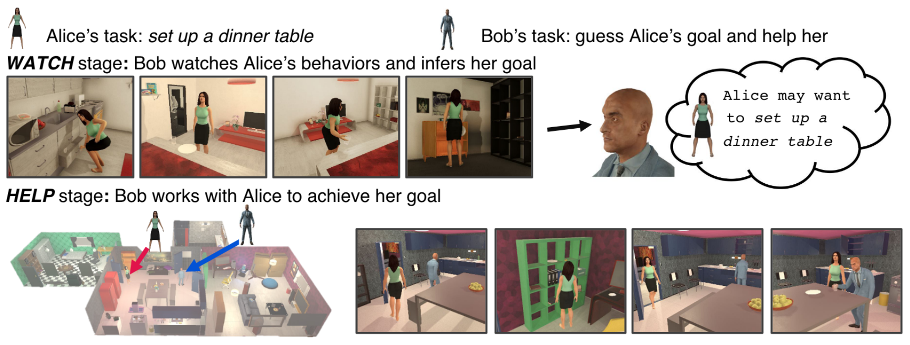
Alice's task: *set up a dinner table*
Bob's task: guess Alice's goal and help her
WATCH stage**:** Bob watches Alice's behaviors and infers her goal
Alice may want to *set up a dinner table*
HELP stage**:** Bob works with Alice to achieve her goal

Figure 12: Watch and Help environment. *source: Figure 1 of (Puig et al., 2021)*

site team which needs to seek them. Besides the overall score, the authors present a metric using transfer into another task. They constructed five *intelligence tests* in the same domain (same action space, same objects, etc.), which test what the agents remembers and what particular abilities they gained. An environment encourages the interesting evolution of team strategies (Figure 13).

MeltingPot MeltingPot (Leibo et al., 2021) is a suit of many games which has customizable map and background population. They have cooperative and mixed settings, further divided into those where prevails cooperation or competition. The main games are CleanUp, Common Harvest, Prisoners' Dilemma, Territory, Overcooked, Stag Hunt, and Chemistry(see Figure 14). Many games have both cooperative and competitive variants. We will offer a very short presentation of all included games, but we refer the reader to the original paper for a more detailed analysis.

CleanUp and **CommonHarvest** variants explore the economical problem known as a *tragedy of the commons* (Hardin, 1968). The agent is motivated to gather fruit because it causes an immediate reward. If exploited selfishly, agents will exhaust the environment and no other fruit will be available, so there is a need to e.g., clean the river once in a while or not to eat the last fruit in some area. Two variants of **Prisoner's Dilemma** represent agent's decision to defect or cooperate by picking respective tokens in the map and they use them if another agent is encountered. The same mechanism is used in **Running with Scissors** games. **Territory** game goal is to gain own territory, but there is also a possibility to destroy it for ev-

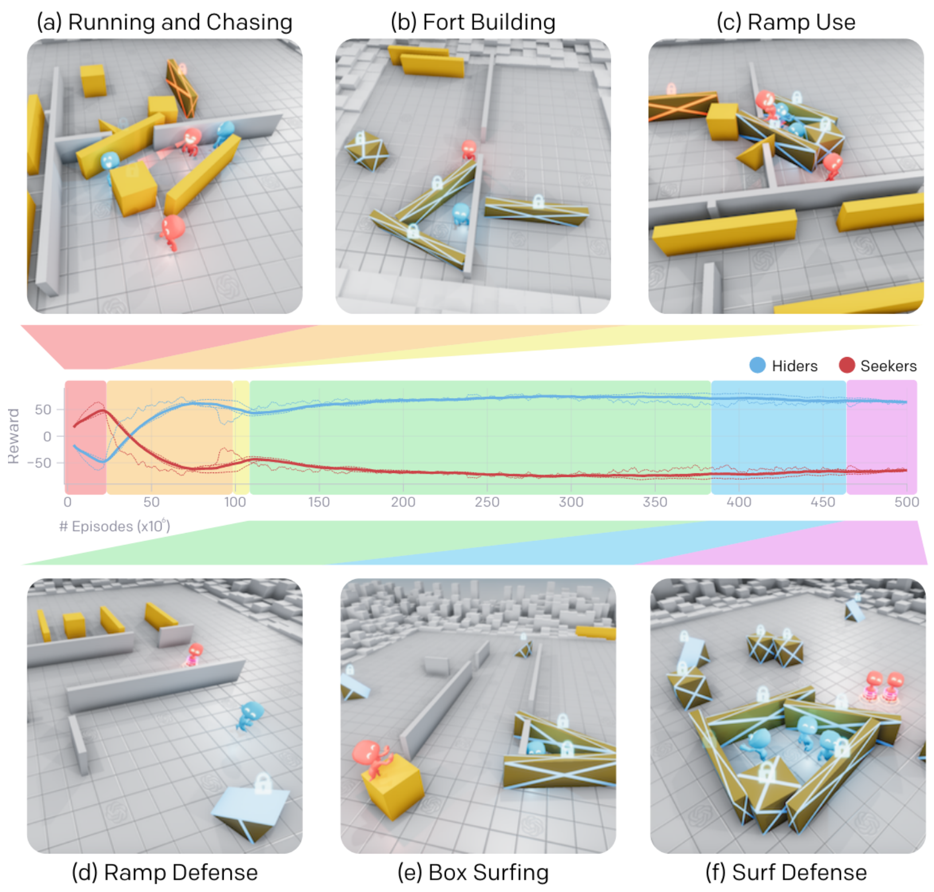

Figure 13: Hide and Seek environment. This figure illustrates 6 different strategy stages which developed during training. Authors argue that this environment uses the same natural dynamics of rivalry that forces evolution. *source: Figure 1 of (Baker et al., 2019)*

eryone. Team variants of territory claiming are **Capture the Flag** and **King of the Hill**. For more cooperative settings, MeltingPot implements Overcooked, Pure Coordination (game of agreeing on the same color), cooperating on producing the right chemical reactions in **Chemistry**, and a complex variant of Rousseauss's **Stag Hunt** game (Rousseau, 1984) known from the game theory.

PettingZoo PettingZoo (Terry et al., 2020a) is an multi-agent equivalent of popular single-agent RL environment Gym (Brockman et al., 2016). PettingZoo contains 61 games divided into 6 categories (number of games included): Ata ri(24), Classic(15), Butterfly(4), MAgent(6), MPE(9), and SISL(3) (see Figure 15).

Pommermman Pommeman (Resnick et al., 2018) is an environment based on the game Bomberman. The main game action is to place a bomb in such a way, it will hurt only the agent's enemies. Pommerman contains both cooperative team variants and competitive scenarios for at least 4 players and is also suitable for exploration of ad-hoc settings.

Others Most of the environments provide either a visual or internal representation of the world. (Park, Oh, and Lee, 2021), on the contrary, wants to provide a very realistic human experience of sensing for training human-like agents, so this environment provides audio, visual, tactile, and proprioceptional perceptions. Another bunch of environments represents very specific abilities like vision understanding, surviving, or in-door scenes (Song et al., 2020). Other environments includes Apprentice Fireman Game (Palmer, Savani, and Tuyls, 2019), variants of Cooperative Multi-agent Object Transportation Problem, NeuralMMO (Suarez et al., 2019), MARLO (Perez-Liebana et al., 2019), and others.

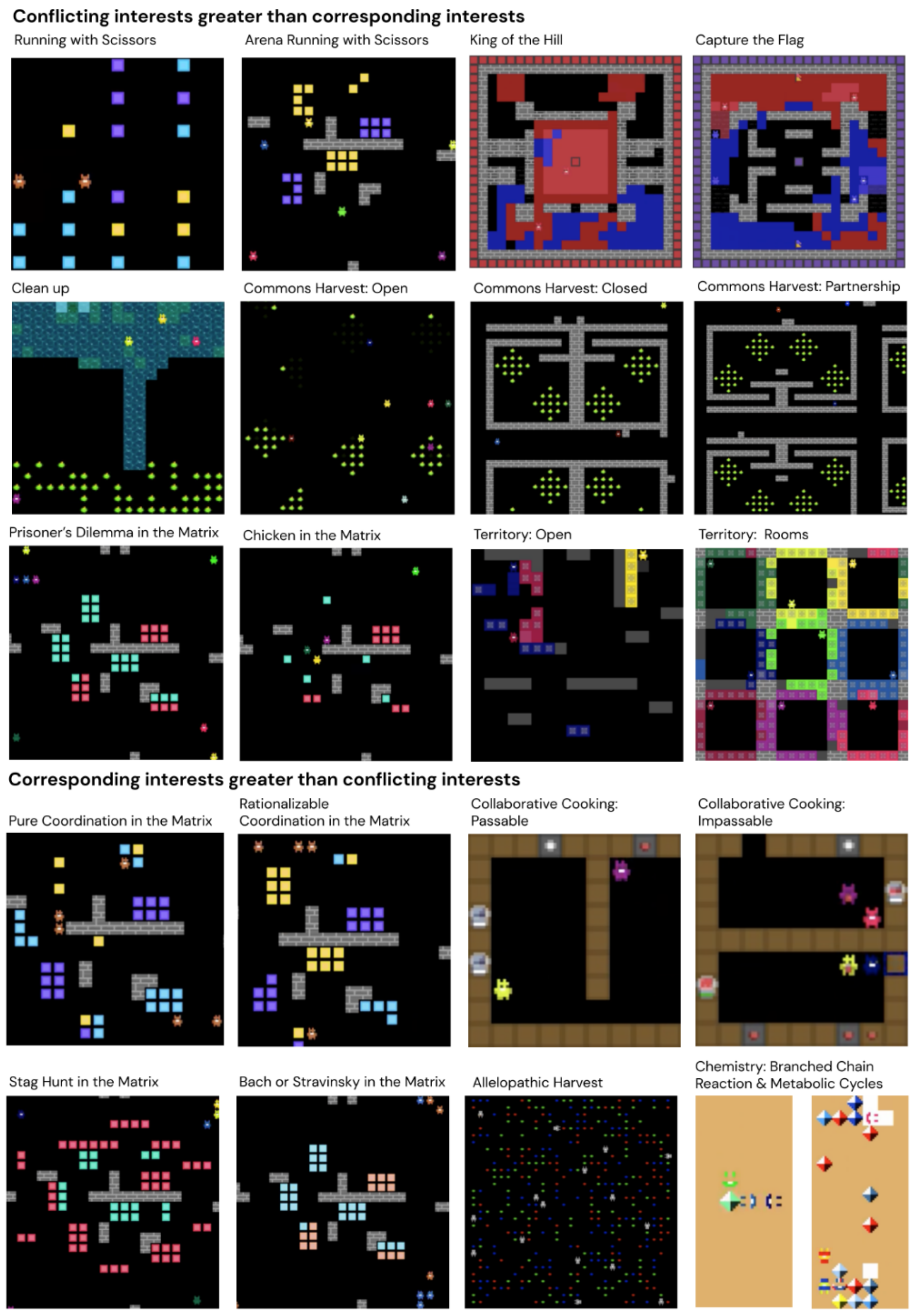
Conflicting interests greater than corresponding interests
Running with Scissors
Arena Running with Scissors
King of the Hill
Capture the Flag
Clean up
Commons Harvest: Open
Commons Harvest: Closed
Commons Harvest: Partnership
Prisoner's Dilemma in the Matrix
Chicken in the Matrix
Territory: Open
Territory: Rooms
Corresponding interests greater than conflicting interests
Pure Coordination in the Matrix
Rationalizable Coordination in the Matrix
Collaborative Cooking: Passable
Collaborative Cooking: Impassable
Stag Hunt in the Matrix
Bach or Stravinsky in the Matrix
Allelopathic Harvest
Chemistry: Branched Chain Reaction & Metabolic Cycles

Figure 14: An overview of all 20 substrates of MeltingPot. *source: Figure 1 of (Leibo et al., 2021)*

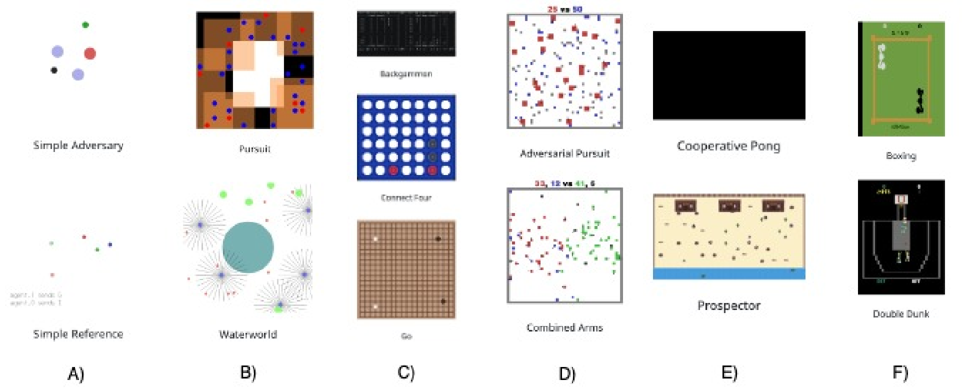

Figure 15: Examples of PettingZoo games. A) Multi Particle Environment (MPE) contains good (green) and bad (red) individuals and allows communication, moving, pushing, etc. B) SISL are three cooperative tasks taken from (Gupta, Egorov, and Kochenderfer, 2017). C) Classical board and card games. D) MAgent is configurable and contains a big amount of particle agents. Originally here: `https://github.com/geek-ai/MAgent` E) Butterfly are visual games that strongly require cooperation. F) Set of Atari games.

4 (Why) We Need Adaptive and Social Agents

In this section, we will try to describe why adaptivity and awareness of others are important for MAS, we will revise commonly researched MARL goals from this perspective and present an overview of the current work. These two abilities are necessary not only for cooperation in general but are also useful for many other MAS problems studied separately from each other. Here is the list of some areas, where adaptability and awareness are the keys, but surely more examples exist, and that is why we see these two abilities to be so important:

- Human-AI interaction (also denoted human-robot interaction (HRI),
- New games with old team-mates (denoted NG-goal)
- Old games with new (co)-players (denoted NP-goal),
- Heterogeneous groups of agents,
- The ability to take on different roles,
- Game-theoretic properties.

More general and complex MARL goals taxonomy can be found in (Wong et al., 2021), where authors distinguish between modelling, communication, efficient cooperation, reward shaping, and quicker training as directions of possible improvement for future research.

We will now describe how awareness and adaptability are useful for many of mentioned tasks, and after that, we will explain more about awareness and adaptability on their own.

4.1 Cooperation & Game-Theoretical Properties

One of the basic goals of MARL is 'just' to solve the task successfully, e.g., train a team of agents to cooperate on solving it. This problem consists of two sub-problems: How to create agents who *want to* cooperate and how to *improve* cooperation if agents want to cooperate. We have described basic MARL algorithms in the section 2.4. For our survey, we would like to emphasize methods beyond self-play and focus on more specific goals, which are currently out of self-play algorithms' power. For deeper survey of MARL algorithms, we refer readers to general MARL surveys (Buşoniu, Babuška, and De Schutter, 2008; Du and Ding, 2020; Gronauer and Diepold, 2022; Hernandez-Leal et al., 2019; Padakandla, 2021). Some papers focus on convergence to Nash equilibria (Bakhtin et al., 2021; Bansal et al., 2020; Hu and Wellman, 2003) or finding best response, stability (Buşoniu, Babuška, and De Schutter, 2008) etc. While solving these things, problems along the way are how to break ties equally for all players, how to choose the same equilibrium, and so on, which can be solved via human-defined conventions, emergent norms, or using communication. One approach which takes others into account is LOLA (Foerster et al., 2018), described more in 6.2, which managed to achieve cooperation where other MARL algorithms[4] failed.

It is also possible to focus mainly on convergence and place the responsibility for quality cooperation upon the benchmark/training environment selection (Fontaine et al., 2021).

4.2 Human-Robot Interaction (HRI)

The important question when it comes to the application of other agent modelling is *Who will the other agents in the environment be?*. It can be other AI agents or humans. Humans differ from other intelligent agents, at least with the current state of knowledge, but it may change with more human-like agents and by moving more towards the general AI. Humans communicate in natural languages and have a lot of knowledge about the world. Although they are able to use more compressed communication protocols, they are definitely less capable when it comes to obtaining and sending information in computer-friendly representation. Human agents are also typically suboptimal and tend to make different decisions, especially choosing differently from equally good equilibria. Humans can also start feeling scared or uncomfortable, but their big advantage is their adaptability. As (Carroll et al., 2019) showed, in the Human-AI interaction, humans typically adapt their behavior according to their team-mates, while AI agents are unable to do so and they need to be the leaders in such interactions. An ability to adapt is therefore very important for better cooperation with humans. Of course, if we had truly adaptive agents, we wouldn't care if they cooperate with humans or robots. While (Carroll et al., 2019) achieved better human cooperation by training them specifically with a human model, (Strouse et al., 2021) trained agents with a diverse group of agents to achieve better adaptability, which they showed to be later useful for Human-AI interaction. To summarise human-AI problem solutions, one direction is to train more adaptive agents (typically using

[4]namely WoLF, policy hill-climbing and JAL-Q

heterogeneous groups and diverse environments) and the other is to train agents specifically for human interaction by training with humans/human models. As for the second option, (McIlroy-Young et al., 2021) aims to achieve a slightly different thing in AI-Human (A-H) interaction: training individual models for each person to recognize and predict her next actions. They achieved very good performance in distinguishing and predicting players in chess, starting with the superhuman chess model AlphaZero (Silver et al., 2018) and prediction Maia model (McIlroy-Young et al., 2020) and achieving very good performance in the personalized modelling of 400 players.

An understanding of others' actions is important not only for cooperation or competition with them but also while simply sharing the environment with others, which indirectly affects them. In (Bansal et al., 2020), they define this as a *parallel play*, giving examples of a shared kitchen in the workplace, or driving on the road with others. They were able to improve HRI using the Bayesian approach to estimating which Nash equilibria will the human converge to. Their approach, however, does not use RL and is poorly scalable. Specifically, understanding people is used for example in the area of the autonomous vehicle, where understanding other drivers and pedestrians (Sadigh et al., 2016; Ziebart et al., 2009) is the key for the save co-existence, usually using inverse RL, which is in (Nikolaidis et al., 2015) joined with the categorization of humans. This can help find the most suitable model personally for each human and leads to a smoother HRI (as was subjectively evaluated by people).

4.3 Understand to Adapt

The important part of being adaptive is understanding others. For example, in one type of kitchen in Overcooked environment there is a need for one-direction circulation movement in the kitchen. It does not matter whether it will be clockwise or counter-clockwise, but all agents should do the same thing. Our ideal adaptive agent should understand, that his team-mate is trying to reach something and that a good solution would be to move in the same direction even if the shortest path to its goal is different. In a more complicated Overcooked setting, it is important to understand what your team-mate is doing. For example, he is preparing tomatoes, so there is no need for chopping tomatoes and better use of time would be to prepare another part of the recipe, e.g., onions. As mentioned earlier, in many situations it is impossible to plan everything in advance either because of the stochastic or partially observable environment, size of the problem, or other agents which we do not know in advance, or we do not control. Understanding others and modifying behavior according to new situations are key skills in such an environment.

4.4 Generality

As we want to utilize all the skills agents may have pursued during training for one task, we would like to transfer them in two situations – taking the whole team of agents to the new environment (NG-goal), and adapting to new team-mates and opponents in the known task (NP-goal).

New games with old pals (NG-goal) The NG-goall describes taking the well-functioning team to solve new problems (potentially in the same domain) and take advantage of a good inter-team cooperation (Shih et al., 2021). We know this use case from the real world, where good teams of professionals perform many new tasks, e.g., army groups, construction, or software companies, fire brigades, etc.

Old games with new co-players or enemies (NP-goal) This task include so-called *ad-hoc* play and *zero-shot* play. These two terms are sometimes used in the same context, although (Hu et al., 2020) defines the first one as trying to adjust policy and convention to the new team (or its part) using agent modelling, adaptability, and so on. In comparison, zero-shot agents do not learn and adapt during the execution and do not bring assumptions about other agents from training time. In other words, they do not *change* the policy during execution, their policy *works* with many various agents, and in the case of (Hu et al., 2020) also demands other agents to be trained for zero-shot. Zero-shot, ad-hoc and adaptive agents are interconnected problems and also many ad-hoc papers do not define adaptive agents in the sense of learning a policy during execution.

One important approach to achieving adaptability to others, and good performance in the heterogeneous group including human players is training agents in a very rich environment: in-domain tasks differing only in the reward, different maps, and a diverse pool of other players (Canaan et al., 2019; Leibo et al., 2021; Strouse et al., 2021)

4.4.1 Models separation

According to (Shih et al., 2021), a good approach for both tasks would be to separate the model for other agents from the model of the environment and of game rules, which can be an important step in allowing more effective learning of both task and agents related skills. Authors do not assume such separation to be a part of input and agents are trained to distinguish which achievement relates to the game specifics and what is a good behavior only in the context of the current group of agents. Model separation is presented also in the sophisticated mind models, e.g. (Rabinowitz et al., 2018) (see section 6.2).

4.5 Heterogeneous groups

The usual technique in MARL is training a group of architecturally identical agents via self-play. What we sometimes want to do is a heterogeneous team. As described in section 2.1 agents can differ in many ways and we may want such different agents to cooperate. Some sources even argue that social diversity leads to mode powerful teams (Kurek and Jaśkowski, 2016), and introducing diversity during training time can also help to improved results in NP-goals or human-AI coordination as described in the previous section. One type of heterogeneous group involves humans, which is explored in more detail before. Other possibilities of heterogeneity in the system are: agents taking different roles, having different architecture, possible actions, etc. Groups can also become more diverse by introducing new players after training (NP-goal).

Adaptive agents would be amazing for all these reasons, yet they introduce additional unpredictability to the system, which would further complicate the problem solving (Hernandez-Leal, Kartal, and Taylor, 2019). It seems to be an inevitable next step in research moving focus to always-changing and adaptive systems, complicated as nature itself. In addition to previously described solutions, we will discuss some attempts which address adaptability and awareness directly in the next section. In (McKee et al., 2020), authors introduces different loss functions to incorporate desired social preferences of the agents by composing the loss function of not only the environment reward (extrinsic) but also with intrinsic motivation term (see picture 16).This study (McKee et al., 2020) also concluded, that heterogeneous groups are both productive and they help to avoid the lazy agent problem. The open question is how to sample the population which will lead to the best performance. We can conclude there are two approaches to deal with the unexpected environment around agents – train them with very diverse surroundings (agents, rewards, environments, etc.) so they are not surprised by the same stochastic nature in the execution time, and learn them to explicitly model thing around them, which will be described in Section 6.2.

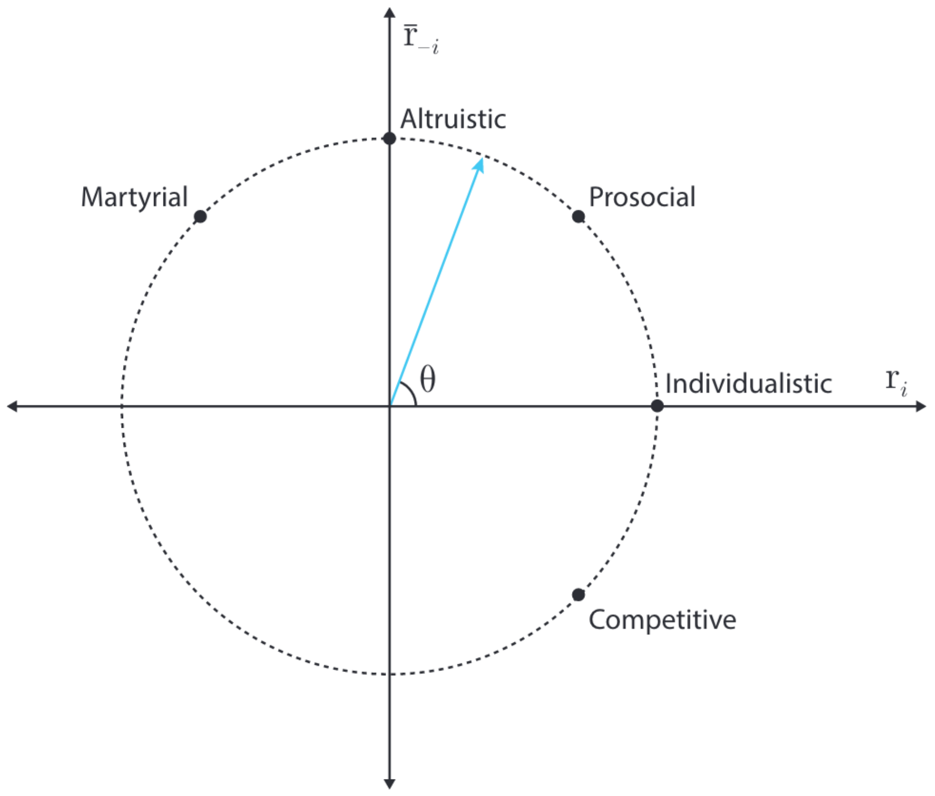

Figure 16: Social Value Orientation proposed in (McKee et al., 2020). Reward angles represent the relationship between personal and others' reward.

4.6 Different Roles

As has been particularly studied in the field of emergent communication, agents are often capable to take only one social role. Examples of such roles are sender and receiver in communication but also leader for convention settings[5] (Carroll et al., 2019; Lewis, 1969).

The same problem of agents being unable to learn different roles, or at least using the same protocol for both roles is present in emergent communication (Lazaridou and Baroni, 2020)

[5]conventions can be defined as *'an arbitrary solution to a recurring coordination problem* (Shih et al., 2021)

5 Adaptability

As described in fig. 17, environment and other agents can change in different ways, we can even consider new players as a change in old players policy. What we call *adaptability* in this text can be actually divided into two

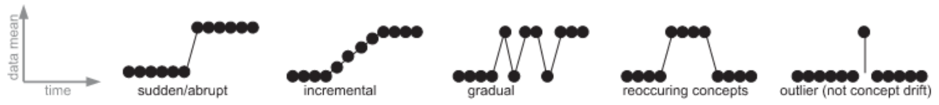

Figure 17: Possible types of overtime changing as described in (Gama et al., 2014).

different abilities. What the majority of papers come up with are agents who are not explicitly adaptive in the sense of continual learning and updating policy during the execution time. They rather learned during training how to react to a diverse set of situations or agent types, so they are not surprised when they encounter the same variety of agents during the execution time (McKee et al., 2020). Another possible approach is an explicitly incorporated adaptability as a part of the agent's architecture and letting the agent learn during execution. This can include explicit modelling of other agents and environment, allowing policy updates during execution, or including the memory of others' behavior. It is also worth noticing, that adaptation and modelling form two steps in the interaction process: first, recognize what the agent wants to do, and second, choose the apropriate policy. The first step can be done using inverse planning or inverse RL, while in the second step we have options of choosing from a fixed set of prepared policies, compute the best response if possible or continuing to learn from new experience.

As mentioned many times in this survey, adaptability is a key ability for problems of both playing familiar games with new players and transferring team capabilities to new tasks (Shih et al., 2021) (NP and NG goals mentioned before).

5.1 Continuous Learning

What we mean by adaptability is a change of behavior according to the situation, namely as a reaction to new players or different environment dynamics. The straightforward idea is just not to quit learning after the training phase and continue updating policy during execution. As it is not a widely used approach, it surely has disadvantages: model guarantees, forgetting, and slow convergence.

Guarantees First problem is that after training we have a model with measured performance, maybe with explainable behavior and other guarantees about its future behavior, which we would simply destroy by additional learning, or the learning process would demand to be also well grounded in theory to preserve the model properties.

Forgetting What we definitely do not want is to forget what our model learned in the training phase. What we want our models to be like for the execution fine tunning is that they would not forget what they learned, just slightly change behavior according to new observations (remember that we typically want to preserve the task or at least stay in the same domain). *Meta-learning* is dealing with solving this problem by selecting models with exactly such properties (Finn, Abbeel, and Levine, 2017). While applying neural networks, there are infinitely many ways how to set the weights in the neural network model which results in the same test data performance. Some of these weights are however on the edge between e.g. classification into a different class and thus further learning can lead to quick change and forgetting. (Finn, Abbeel, and Levine, 2017) tries to learn models which are not so prone to quick radical change.

Slow convergence While it is important to have a model that does not forget too much, we also need a model to learn relatively quickly, so we do not lose the game or bankrupt the company before the model starts to be good again.

5.2 Prepared during Training

Switching policies We need adaptation if the current policy produces much worse results than expected from the training time. For this, we need to detect the change, and also learn an appropriate response (new policy) (Wong et al., 2021). If the change is caused by the non-stationary environments, ideally recurring and slow (Hernandez-Leal et al., 2019), we can use *context detection algorithms* (Da Silva et al., 2006). In this approach, the agent maintains the library of partial models of the environment. After every step, the best suitable model is chosen as the currently used model. If no model is satisfying (the error is above the given threshold), the new model is built. For more detailed survey, see (Padakandla, 2021). We can do the same while detecting explicitly the change in the behaviour of other agents rather than taking them as part of the environment. Some examples of detecting algorithms are MDP-CL (Hernandez-Leal, Cote, and Sucar, 2014; Hernandez-Leal et al., 2016), or DriftEr (Hernandez-Leal et al., 2017). Other important switching algorithms are WOLF-ifa, WOLF-hill climbing, AWESOME (Conitzer and Sandholm, 2007), DRON (He et al., 2016), SAM (Everett and Roberts, 2018), (Padakandla, K. J., and Bhatnagar, 2020) and Deep BPR+ (Zheng et al., 2018). This *policy switching* idea is used also in the game theory, for example in the iterated prisoners dilemma study (Press and Dyson, 2012), where the combination of opponent's actions memory, and policy switching produces interesting strategies (Jurišić, Kermek, and Konecki, 2012). We will discuss the dual method – switching models of other in the next section.

Conventions Conventions are mentioned primarily as a tool for breaking ties in a coordinated manner. Relying upon conventions learned during training can harm the adaptation to new agents, as they can follow different conventions. We can want to avoid conventions at all, which shift us to *zero-shot* setting, or we can encourage them to cooperate in games like Hanabi, or Overcooked (Shih et al., 2021).

Explicit separation of convention from the rest of the model is proposed in (Shih et al., 2021). The model is trained simultaneously on several in-domain tasks (same dynamics, different rewards) and with many different agents to encourage learning of distinction between task-related and agents-related events. Authors also conducted a human study to verify the hypothesis about carrying over conventions developed previously in a similar task, e.g. if actions a, b in state s is optimal in both old and new task, humans continue using the same convention for choosing the action in the new task and this behavior was also achieved by the proposed model.

6 Awareness

The main taxonomy according to awareness is distinction between *agent-aware*, *agent-tracking*, or *agent-independent* (Buşoniu, Babuška, and De Schutter, 2008) agents. In term of other agents' modelling and adaptation, this taxonomy divides algorithms based on whether they explicitly model other agents (agent-tracking) or just learns how to react to different types of interaction, e.g. change of team-mates (agents-aware). As mentioned before, we can pretend that other agents are a part of the environment and use simple single-agent RL methods, which is possible but very difficult (Bard et al., 2020) and other approaches are usually preferred. More detailed descriptions with example algorithms of each category can be found in (Buşoniu, Babuška, and De Schutter, 2008) and we also supply a brief overview.

6.1 Agent-Independent

Agent-independent methods use the game theory (possibly together with RL) to find the best policy and value function in each state. These solutions suffer mainly from the suboptimal solution because of missing coordination. As every agent can theoretically learn its part of a good equilibrium strategy, there is no mechanism to assure all agents will choose the same equilibrium. One possible solution could be to learn norms or define them in advance.

6.2 Agent-Tracking

If we decide to incorporate some explicit model of other agents, we could learn their **q-value**, **goal**, or more complicated **models of mind**.

Q-value This approach basically simulates other agents' decision process from the technical side, thus learning what are their action probabilities in each state.

Goal This approach can be related to the inverse planning (Ng and Russell, 2000), as we want to find out what is other agents' goals. This is easier in an environment with strict task definition and hierarchy, where the goal is to match subtasks and tasks to observed behavior, e.g. recipes in Overcooked (Wang et al., 2020).

In (Wang et al., 2020), agents use inverse planning to estimate task allocations (= who is doing what). If they agree upon the same task allocation, there is no problem in high-level cooperation and they only need to solve low-level collision avoidance. As they cannot communicate, they are only estimating others' chosen task allocations and trying to reach conformity. Reinforcement learning is used for the best action selection after deciding the most probable task allocation, while (Foerster et al., 2018) presents the possibility of reasoning about others on different levels.

Besides RL, we could also use classical planning together with plan recognition.

Limited Modelling What we call here a *limited modelling* are simple models, which, in contrast to models in the next paragraph, do not explicitly learn some complicated architecture of other agents, but rather categorize others. As discussed earlier, the second part of the solution must be choosing the right response policy. The most basic approach is to assume others to be either friends or enemies (Littman, 2001), or better be prepared that other agents can be drawn from a predefined set of characters (Nikolaidis et al., 2015, 2016). This approach has obvious limitations depending on the advanced knowledge of all possibilities. An interesting generalization of this modelling can be found in (Foerster et al., 2018) Learning Opponent-Learning Awareness (LOLA). LOLA method adds one additional term to the update rule and it is applicable to gradient methods. The additional term express how the policy change will affect the other agent's learning. Although the basic version (first-order LOLA) assumes other agents to be what they call naive learners (= others do not use LOLA and do not take us using LOLA into account), it is way more general than the previously described categorization. This algorithm was primarily developed to achieve cooperation, which works great in comparison to many other MARL algorithms benchmarking on Iterated Prisoners Dilemma, Iterated Matching Pennies, and Coin Game. The paper also mentions higher-order LOLA, when we assume our opponent to use LOLA also, or that he assumes that we assume that he uses LOLA, etc. This is the concept known also from the economy as a *bounded rationality* (Mehta, Starmer, and Sugden, 1994). In some situations, there is a need to make a choice, but there is no rational reason for any particular option. In such a situation, people are making decisions based on many psychological processes, but these processes can not be described mathematically, therefore we call them *nonrational*. If we are however making our choices in these situations by reasoning about others in higher orders, we are taking some reasons into account to prioritize some actions over others, thus the term *bounded rationality*.

We can extend this categorization approach in two directions: using a memory of others' past actions, or modeling others more sophisticatedly. We will first describe the memory usage here, a mind models description will follow.

If we are speaking about memory, we can have *unbounded* or *memory-bounded* agents. Unbounded agents take into account the whole interaction history, e.g. (Foerster et al., 2018), while memory-bounded agents use just a fixed-size window of history (Chakraborty and Stone, 2014) – the extreme example are decisions based only on the last player's action.

Mind Models Unlike the previous approach, here the goal is to create models inspired by how we imagine the human mind is working. This includes the most complicated approximations of other agents, using concepts such as motivation, reliability, beliefs, selfishness, etc. In psychology, this reasoning about others' behavior is called *theory of mind*. It is also possible to model just some of these properties, one important way, in particular, is to model what other beliefs about the environment, as they can be (mis) led by (potentially false) beliefs about the world, they would have different pieces of information than our agent has, or possibly they are unable to know something we observed.

One of the first applications of the theory of mind to AI is ToMNet (Rabinowitz et al., 2018), a deep neural model which has explicit separate networks for modelling an agent's character, current mental state, and the prediction of future agent's behavior.

Tracking public beliefs is not a very general approach, because it demands knowing the game mechanics, and methods usually do not scale well, but in small games of two players, they can present an interesting point of view to MAS problems. As examples, we would like to mention BAD (Foerster et al., 2019) and (Chalkiadakis and Boutilier, 2003) which both use the Bayesian approach.

6.3 Agent-Aware

Agent-aware models focus on the convergence to equilibria while adapting to other agents, whom it does not ignore but also does not model explicitly. We want to somehow differ from the classical single-agent algorithms as we know, that there are other learning and acting agents, but rather without an explicit model of them. One simple possibility is just as ordinary learning the best action. In this section, we can again mention policy-switching methods (see 5.2), mainly AWESOME and Win-or-Learn-Fast modifications.

7 Conclusion

We reviewed existing problems, benchmarks, and interesting approaches in multi-agent systems with an emphasis on cooperation, adaptability, and awareness of other agents. As we showed, there is an important connection between adaptability, awareness, and other partial MAS tasks. Research in MAS with the use of reinforcement learning is conducted in many areas and possible benchmarks reflect this variance. There is a quickly rising importance of studying problems in a multi-agent context and reinforcement learning has even greater potential there, because of a size, complexity, unpredictability, and a need for quick adaptation in the continuously unexpectedly changing world. As MARL is currently a fast-growing research field, it is somewhat fragmented. There is definitely a need for better comparability between algorithms. We see a big opportunity for future work in

- thorough analysis of flaws, specifics, and advantages of existing approach not only for one task but across different areas,
- research focused on more general goals of adaptive agents, who can model others, although special task definitions, e.g. ad-hoc or zero-shot play, communication, heterogeneity, or Human-AI interaction can serve well for developing such general agents,
- extension of the existing approach to more complicated situations (as the majority of methods were tested on two-player scenarios only).

Another possible direction is incorporating communication, which can be useful for:

- accelerating adaptation to new agents (Zhu, Neubig, and Bisk, 2021),
- sharing information, if agents have different knowledge,
- communicating goals explicitly,
- agreeing on next steps
- exploring the possibility of methods transfer between communication and others' modelling.

Glossary

%DET %Determinism. 19

EGT evolutionary game theory. 16, 17

JRPs joint recurrence plot. 19

MARL multi-agent reinforcement rearning. 1, 11

MAS multi-agent system. 1, 2, 11

RL reinforcement learning. 1, 2, 4, 6, 7, 9–12

RQA recurrence quantification analysis. 19

SARL Singe-Agent Reinforcement Learning. 7, 11

TD temporal difference. 9

WoLF Winf-or-Learn-Fast. 13

References

Baffier, Jean-François et al. (2017). "Hanabi Is NP-hard, Even for Cheaters Who Look at Their Cards". In: *Theoretical Computer Science* 675, pp. 43–55. ISSN: 0304-3975. DOI: 10.1016/j.tcs.2017.02.024. URL: https://www.sciencedirect.com/science/article/pii/S0304397517301573.

Baker, Bowen et al. (2019). "Emergent Tool Use from Multi-Agent Autocurricula". URL: https://arxiv.org/abs/1909.07528 (visited on 10/12/2022).

Bakhtin, Anton et al. (2021). "No-Press Diplomacy from Scratch". In: *Advances in Neural Information Processing Systems*. Advances in Neural Information Processing Systems. Vol. 34. Curran Associates, Inc., pp. 18063–18074. URL: https://proceedings.neurips.cc/paper/2021/hash/95f2b84de5660ddf45c8a34933a2e66f-Abstract.html (visited on 07/20/2022).

Bansal, Shray et al. (June 10, 2020). "A Bayesian Framework for Nash Equilibrium Inference in Human-Robot Parallel Play". URL: http://arxiv.org/abs/2006.05729 (visited on 05/27/2022).

Bard, Nolan et al. (Mar. 1, 2020). "The Hanabi Challenge: A New Frontier for AI Research". In: *Artificial Intelligence* 280, p. 103216. ISSN: 0004-3702. DOI: 10.1016/J.ARTINT.2019.103216.

Beck, Jacob et al. (2019). "Amrl: Aggregated Memory for Reinforcement Learning". In: *International Conference on Learning Representations*. International Conference on Learning Representations. Vol. 8th. Addis Ababa, Ethiopia.

Bellman, Richard (1966). "Dynamic Programming". In: *Science* 153.3731, pp. 34–37.

Bloembergen, Daan et al. (Aug. 17, 2015). "Evolutionary Dynamics of Multi-Agent Learning: A Survey". In: *Journal of Artificial Intelligence Research* 53, pp. 659–697. ISSN: 1076-9757. DOI: 10.1613/jair.4818. URL: https://www.jair.org/index.php/jair/article/view/10952 (visited on 08/01/2022).

Blum, Avrim and Yishay Mansour (2007). "Learning, Regret Minimization, and Equilibria". In: *Algorithmic Game Theory*. Ed. by Eva Tardos et al. Cambridge: Cambridge University Press, pp. 79–102. ISBN: 978-0-521-87282-9. DOI: 10.1017/CBO9780511800481.006. URL: https://www.cambridge.org/core/books/algorithmic-game-theory/learning-regret-minimization-and-equilibria/46547C9BEFE2920814A887AC5BABDDCB (visited on 10/12/2022).

Boutilier, Craig (1996). "Planning, Learning and Coordination in Multiagent Decision Processes". In: *In Proceedings of the Sixth Conference on Theoretical Aspects of Rationality and Knowledge (TARK96*. Conference on Theoretical Aspects of Rationality and Knowledge. Vol. 6, pp. 195–210.

Bouzy, Bruno (2017). "Playing Hanabi Near-Optimally". In: *Advances in Computer Games*. Springer, pp. 51–62. ISBN: 978-3-319-71648-0. DOI: 10.1007/978-3-319-71649-7_5.

Bowling, Michael and Manuela Veloso (2001). "Rational and Convergent Learning in Stochastic Games". In: *International Joint Conference on Artificial Intelligence*. Vol. 17. 1, pp. 1021–1026. ISBN: 978-1-4471-5102-9. DOI: 10.1007/978-1-4471-5102-9_33-3.

Bowling, Michael et al. (Jan. 9, 2015). "Heads-up Limit Hold'em Poker Is Solved". In: *Science* 347.6218, pp. 145–149. DOI: 10.1126/science.1259433. URL: https://www.science.org/doi/10.1126/science.1259433 (visited on 02/15/2022).

Brandt, Carsten (2020). "Recurrence Quantification Compared to Fourier Analysis for Ultrasonic Non-Destructive Testing of Carbon Fibre Reinforced Polymers". PhD thesis. University of Bremen. 278 pp. URL: https://www.researchgate.net/profile/Carsten-Brandt/publication/

342945957_Recurrence_Quantification_Compared_to_Fourier_Analysis_for_Ultrasonic_Non-Destructive_Testing_of_Carbon_Fibre_Reinforced_Polymers/links/5f0eb985299bf1e548b6ff9c/Recurrence-Quantification-Compared-to-Fourier-Analysis-for-Ultrasonic-Non-Destructive-Testing-of-Carbon-Fibre-Reinforced-Polymers.pdf (visited on 12/10/2022).

Brockman, Greg et al. (2016). "Openai gym". In: *arXiv preprint arXiv:1606.01540*.

Buşoniu, Lucian, Robert Babuška, and Bart De Schutter (2008). "A Comprehensive Survey of Multiagent Reinforcement Learning". In: *IEEE Transactions on Systems, Man and Cybernetics Part C: Applications and Reviews* 38.2, pp. 156–172. ISSN: 10946977. DOI: 10.1109/TSMCC.2007.913919.

Börgers, Tilman and Rajiv Sarin (1997). "Learning through Reinforcement and Replicator Dynamics". In: *Journal of economic theory* 77.1, pp. 1–14.

Campbell, Murray, A. Joseph Hoane Jr, and Feng-hsiung Hsu (2002). "Deep Blue". In: *Artificial intelligence* 134.1-2, pp. 57–83.

Canaan, Rodrigo et al. (Aug. 2019). "Diverse Agents for Ad-Hoc Cooperation in Hanabi". In: *2019 IEEE Conference on Games (CoG)*. 2019 IEEE Conference on Games (CoG). London, United Kingdom: IEEE, pp. 1–8. ISBN: 978-1-72811-884-0. DOI: 10.1109/CIG.2019.8847944. URL: https://ieeexplore.ieee.org/document/8847944/ (visited on 10/12/2022).

Carroll, Micah et al. (Dec. 8, 2019). "On the Utility of Learning about Humans for Human-AI Coordination". In: *Proceedings of the 33rd International Conference on Neural Information Processing Systems*. 465. Red Hook, NY, USA: Curran Associates Inc., pp. 5174–5185.

Chakraborty, Doran and Peter Stone (Mar. 1, 2014). "Multiagent Learning in the Presence of Memory-Bounded Agents". In: *Autonomous Agents and Multi-Agent Systems* 28.2, pp. 182–213. ISSN: 1573-7454. DOI: 10.1007/s10458-013-9222-4. URL: https://doi.org/10.1007/s10458-013-9222-4 (visited on 09/20/2022).

Chalkiadakis, Georgios and Craig Boutilier (2003). "Coordination in Multiagent Reinforcement Learning: A Bayesian Approach". In: *Proceedings of the Second International Joint Conference on Autonomous Agents and Multiagent Systems*, pp. 709–716. DOI: 10.1145/860575.860689.

Chen, Ruizhi et al. (Sept. 3, 2021). *Eden: A Unified Environment Framework for Booming Reinforcement Learning Algorithms*. DOI: 10.48550/arXiv.2109.01768. arXiv: 2109.01768 [cs]. URL: http://arxiv.org/abs/2109.01768 (visited on 08/11/2022).

Conitzer, Vincent and Tuomas Sandholm (May 2007). "AWESOME: A General Multiagent Learning Algorithm That Converges in Self-Play and Learns a Best Response against Stationary Opponents". In: *Machine Learn-*

ing 67.1-2, pp. 23–43. ISSN: 0885-6125, 1573-0565. DOI: 10.1007/s10994-006-0143-1. URL: http://link.springer.com/10.1007/s10994-006-0143-1 (visited on 07/12/2022).

Coulom, Rémi (2006). "Efficient Selectivity and Backup Operators in Monte-Carlo Tree Search". In: *International Conference on Computers and Games*. Springer, pp. 72–83. DOI: 10.1007/978-3-540-75538-8_7.

Da Silva, Bruno C. et al. (2006). "Dealing with Non-Stationary Environments Using Context Detection". In: *Proceedings of the 23rd International Conference on Machine Learning*, pp. 217–224. ISBN: 1-59593-383-2. DOI: 10.1145/1143844.1143872.

Dafoe, Allan et al. (Dec. 15, 2020). "Open Problems in Cooperative AI". URL: https://arxiv.org/abs/2012.08630v1 (visited on 10/02/2021).

Damani, Mehul et al. (2021). "PRIMAL$_2$: Pathfinding via Reinforcement and Imitation Multi-Agent Learning-Lifelong". In: *IEEE Robotics and Automation Letters* 6.2, pp. 2666–2673. DOI: 10.1109/LRA.2021.3062803.

Deng, Li (2012). "The MNIST database of handwritten digit images for machine learning research". In: *IEEE Signal Processing Magazine* 29.6, pp. 141–142. DOI: 10.1109/MSP.2012.2211477.

Devlin, Jacob et al. (2019). "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *NAACL HLT 2019 - 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies - Proceedings of the Conference*. Vol. 1. Association for Computational Linguistics (ACL), pp. 4171–4186. ISBN: 978-1-950737-13-0. DOI: 10.18653/v1/N19-1423. arXiv: 1810.04805.

Du, Wei and Shifei Ding (Nov. 24, 2020). "A Survey on Multi-Agent Deep Reinforcement Learning: From the Perspective of Challenges and Applications". In: *Artificial Intelligence Review 2020 54:5* 54.5, pp. 3215–3238. ISSN: 1573-7462. DOI: 10.1007/S10462-020-09938-Y. URL: https://link.springer.com/article/10.1007/s10462-020-09938-y (visited on 10/19/2021).

Everett, Richard and Stephen Roberts (2018). "Learning against Non-Stationary Agents with Opponent Modelling and Deep Reinforcement Learning". In: *2018 AAAI Spring Symposium Series*.

Farquhar, Gregory et al. (2018). "TreeQN and ATreeC: Differentiable Tree-Structured Models for Deep Reinforcement Learning". In: *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net. URL: https://openreview.net/forum?id=H1dh6Ax0Z (visited on 10/17/2022).

Fickinger, Arnaud et al. (2021). "Scalable Online Planning via Reinforcement Learning Fine-Tuning". In: *Advances in Neural Information Processing Systems* 34, pp. 16951–16963.

Finn, Chelsea, Pieter Abbeel, and Sergey Levine (July 18, 2017). "Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks". In: *International Conference on Machine Learning*. Vol. 70, pp. 1126–1135.

Foerster, Jakob et al. (Dec. 14, 2017). "Counterfactual Multi-Agent Policy Gradients". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 32. ISBN: 978-1-57735-800-8.

Foerster, Jakob et al. (May 24, 2019). "Bayesian Action Decoder for Deep Multi-Agent Reinforcement Learning". In: *International Conference on Machine Learning*, pp. 1942–1951. ISSN: 2640-3498. URL: `https://proceedings.mlr.press/v97/foerster19a.html` (visited on 10/24/2021).

Foerster, Jakob N. et al. (Sept. 19, 2018). "Learning with Opponent-Learning Awareness". In: *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*. International Foundation for Autonomous Agents and Multiagent Systems, pp. 122–130.

Fontaine, Matthew C. and Stefanos Nikolaidis (July 15, 2021). "Evaluating Human-Robot Interaction Algorithms in Shared Autonomy via Quality Diversity Scenario Generation". In: *ACM Transactions on Human-Robot Interaction*. DOI: `10.1145/3476412`. URL: `https://doi.org/10.1145/3476412` (visited on 08/10/2022).

Fontaine, Matthew C. et al. (2021). "On the Importance of Environments in Human-Robot Coordination". In: Robotics: Science and Systems XVII, Virtual Event, July 12-16, 2021. ISBN: 978-0-9923747-7-8. arXiv: `2106.10853`. URL: `http://www.roboticsproceedings.org/rss17/p038.html` (visited on 04/19/2022).

Fudenberg, Drew and Jean Tirole (1991). *Game Theory*. MIT press.

Fujimoto, Scott, Herke Hoof, and David Meger (2018). "Addressing Function Approximation Error in Actor-Critic Methods". In: *International Conference on Machine Learning*. PMLR, pp. 1587–1596.

Gama, João et al. (2014). "A Survey on Concept Drift Adaptation". In: *ACM computing surveys (CSUR)* 46.4, pp. 1–37.

Gemici, Mevlana et al. (2017). "Generative Temporal Models with Memory". URL: `http://arxiv.org/abs/1702.04649` (visited on 10/13/2022).

Gronauer, Sven and Klaus Diepold (Feb. 1, 2022). "Multi-Agent Deep Reinforcement Learning: A Survey". In: *Artificial Intelligence Review* 55.2, pp. 895–943. ISSN: 1573-7462. DOI: `10.1007/s10462-021-09996-w`. URL: `https://doi.org/10.1007/s10462-021-09996-w` (visited on 08/11/2022).

Gupta, Jayesh K., Maxim Egorov, and Mykel Kochenderfer (2017). "Cooperative Multi-Agent Control Using Deep Reinforcement Learning". In: *International Conference on Autonomous Agents and Multiagent Systems*. Springer, pp. 66–83. ISBN: 978-3-319-71681-7. DOI: 10.1007/978-3-319-71682-4_5.

Hansen, Eric A., Daniel S. Bernstein, and Shlomo Zilberstein (2004). "Dynamic Programming for Partially Observable Stochastic Games". In: *AAAI*. Vol. 4, pp. 709–715. ISBN: 0-262-51183-5.

Hardin, Garrett (1968). "The Tragedy of the Commons". In: *Science* 162.3859, pp. 1243–1248. ISSN: 0036-8075. JSTOR: 1724745.

He, He et al. (2016). "Opponent Modeling in Deep Reinforcement Learning". In: *International Conference on Machine Learning*. JMLR.org, pp. 1804–1813.

Hernandez-Leal, Pablo, Enrique Munoz de Cote, and L. Enrique Sucar (2014). "A Framework for Learning and Planning against Switching Strategies in Repeated Games". In: *Connection Science* 26.2, pp. 103–122. DOI: 10.1080/09540091.2014.885294.

Hernandez-Leal, Pablo, Bilal Kartal, and Matthew E. Taylor (Oct. 16, 2019). "A Survey and Critique of Multiagent Deep Reinforcement Learning". In: *Autonomous Agents and Multi-Agent Systems 2019 33:6* 33.6, pp. 750–797. ISSN: 1573-7454. DOI: 10.1007/S10458-019-09421-1. URL: https://link.springer.com/article/10.1007/s10458-019-09421-1 (visited on 10/21/2021).

Hernandez-Leal, Pablo et al. (2016). "A Bayesian Approach for Learning and Tracking Switching, Non-Stationary Opponents". In: *Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems*, pp. 1315–1316.

Hernandez-Leal, Pablo et al. (2017). "Efficiently Detecting Switches against Non-Stationary Opponents". In: *Autonomous Agents and Multi-Agent Systems* 31.4, pp. 767–789. DOI: 10.1007/s10458-016-9352-6.

Hernandez-Leal, Pablo et al. (Mar. 11, 2019). *A Survey of Learning in Multiagent Environments: Dealing with Non-Stationarity*. arXiv: 1707.09183 [cs]. URL: http://arxiv.org/abs/1707.09183 (visited on 07/12/2022).

Hessel, Matteo et al. (Oct. 6, 2017). "Rainbow: Combining Improvements in Deep Reinforcement Learning". In: *Thirty-Second AAAI Conference on Artificial Intelligence*. arXiv. DOI: 10.48550/arXiv.1710.02298. arXiv: 1710.02298 [cs]. URL: http://arxiv.org/abs/1710.02298 (visited on 08/25/2022).

Hofer, Ronald, Edward Ramirez, and Scott Smith (Jan. 1, 1998). “Automated Scenario Generation Environment”. In: *Institute for Simulation and Training*. URL: https://stars.library.ucf.edu/istlibrary/31.

Hu, Hengyuan et al. (2020). “"Other-Play" for Zero-Shot Coordination”. In: *International Conference on Machine Learning*, pp. 4399–4410.

Hu, Junling and Michael P. Wellman (2003). “Nash Q-learning for General-Sum Stochastic Games”. In: *Journal of machine learning research* 4 (Nov), pp. 1039–1069.

Johanson, Michael (Mar. 7, 2013). *Measuring the Size of Large No-Limit Poker Games*. DOI: 10.48550/arXiv.1302.7008. arXiv: 1302.7008 [cs]. URL: http://arxiv.org/abs/1302.7008 (visited on 10/14/2022).

Jurišić, Marko, Dragutin Kermek, and Mladen Konecki (May 2012). “A Review of Iterated Prisoner’s Dilemma Strategies”. In: *2012 Proceedings of the 35th International Convention MIPRO*. 2012 Proceedings of the 35th International Convention MIPRO, pp. 1093–1097.

Kaiser, Lukasz et al. (Feb. 19, 2020). *Model-Based Reinforcement Learning for Atari*. DOI: 10.48550/arXiv.1903.00374. arXiv: 1903.00374 [cs, stat]. URL: http://arxiv.org/abs/1903.00374 (visited on 07/31/2022).

Kianercy, Ardeshir and Aram Galstyan (2012). “Dynamics of Boltzmann Q Learning in Two-Player Two-Action Games”. In: *Physical Review E* 85.4, p. 041145.

Kim, Nayoung and Chang Nam (Feb. 28, 2020). “Adaptive Control of Thought-Rational (ACT-R): Applying a Cognitive Architecture to Neuroergonomics”. In: pp. 105–114. ISBN: 978-3-030-34783-3. DOI: 10.1007/978-3-030-34784-0_6.

Kraemer, Landon and Bikramjit Banerjee (2016). “Multi-Agent Reinforcement Learning as a Rehearsal for Decentralized Planning”. In: *Neurocomputing* 190, pp. 82–94.

Kröse, Ben J. A. (1995). “Learning from Delayed Rewards”. In: *Robotics Auton. Syst.* 15.4, pp. 233–235. DOI: 10.1016/0921-8890(95)00026-C.

Kurek, Mateusz and Wojciech Jaśkowski (2016). “Heterogeneous Team Deep Q-learning in Low-Dimensional Multi-Agent Environments”. In: *2016 IEEE Conference on Computational Intelligence and Games (CIG)*. IEEE, pp. 1–8.

Laurent, Florian et al. (Mar. 30, 2021). *Flatland Competition 2020: MAPF and MARL for Efficient Train Coordination on a Grid World*. DOI: 10.48550/arXiv.2103.16511. arXiv: 2103.16511 [cs]. URL: http://arxiv.org/abs/2103.16511 (visited on 08/12/2022).

Lazaridou, Angeliki and Marco Baroni (2020). “Emergent Multi-Agent Communication in the Deep Learning Era”.

Leibo, Joel Z. et al. (July 1, 2021). "Scalable Evaluation of Multi-Agent Reinforcement Learning with Melting Pot". In: *Proceedings of the 38th International Conference on Machine Learning*. International Conference on Machine Learning. PMLR, pp. 6187–6199. URL: `https://proceedings.mlr.press/v139/leibo21a.html` (visited on 05/27/2022).

Lewis, David (1969). *Convention: A Philosophical Study*. Cambridge, MA, USA: Wiley-Blackwell.

Lillicrap, Timothy P. et al. (July 5, 2019). *Continuous Control with Deep Reinforcement Learning*. Version 2. arXiv: `1509.02971 [cs, stat]`. URL: `http://arxiv.org/abs/1509.02971` (visited on 08/25/2022).

Littman, Michael L. (1994). "Markov Games as a Framework for Multi-Agent Reinforcement Learning". In: *Machine Learning Proceedings 1994*. Elsevier, pp. 157–163.

— (2001). "Friend-or-Foe Q-learning in General-Sum Games". In: *ICML*. International Conference on Machine Learning. Vol. 1, pp. 322–328.

Ljung, Lennart (2017). "System Identification". In: *Wiley Encyclopedia of Electrical and Electronics Engineering*. John Wiley & Sons, Ltd, pp. 1–19. ISBN: 978-0-471-34608-1. DOI: `10.1002/047134608X.W1046.pub2`. URL: `https://onlinelibrary.wiley.com/doi/abs/10.1002/047134608X.W1046.pub2` (visited on 08/26/2022).

Lowe, Ryan et al. (2017). "Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments". In: *Advances in neural information processing systems* 30.

Marwan, Norbert et al. (2007). "Recurrence Plots for the Analysis of Complex Systems". In: *Physics reports* 438.5-6, pp. 237–329.

Mas-Colell, Andreu, Michael Dennis Whinston, Jerry R Green, et al. (1995). *Microeconomic theory*. Vol. 1. Oxford university press New York.

Matta, Marco et al. (2019). "Q-RTS: A Real-Time Swarm Intelligence Based on Multi-Agent Q-learning". In: *Electronics Letters* 55.10, pp. 589–591.

McIlroy-Young, Reid et al. (2020). "Aligning Superhuman Ai with Human Behavior: Chess as a Model System". In: *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 1677–1687.

McIlroy-Young, Reid et al. (Feb. 13, 2021). "Learning Personalized Models of Human Behavior in Chess". URL: `http://arxiv.org/abs/2008.10086` (visited on 05/27/2022).

McKee, Kevin R. et al. (2020). "Social Diversity and Social Preferences in Mixed-Motive Reinforcement Learning". In: AAMAS 2020. ISBN: 978-1-4503-7518-4. arXiv: `2002.02325`.

Mehta, Judith, Chris Starmer, and Robert Sugden (1994). "The Nature of Salience: An Experimental Investigation of Pure Coordination Games". In: *The American Economic Review* 84.3, pp. 658–673.

Mnih, Volodymyr et al. (Dec. 19, 2013). *Playing Atari with Deep Reinforcement Learning*. arXiv: `1312.5602 [cs]`. URL: `http://arxiv.org/abs/1312.5602` (visited on 08/02/2022).

Mnih, Volodymyr et al. (Feb. 26, 2015). "Human-Level Control through Deep Reinforcement Learning". In: *Nature* 518.7540, pp. 529–533. ISSN: 0028-0836, 1476-4687. DOI: `10.1038/nature14236`. URL: `http://www.nature.com/articles/nature14236` (visited on 02/07/2022).

Mnih, Volodymyr et al. (2016). "Asynchronous Methods for Deep Reinforcement Learning". In: *International Conference on Machine Learning*. PMLR, pp. 1928–1937. arXiv: `1602.01783 [cs]`.

Moerland, Thomas M. et al. (Mar. 31, 2022). *Model-Based Reinforcement Learning: A Survey*. DOI: `10.48550/arXiv.2006.16712`. arXiv: `2006.16712 [cs, stat]`. URL: `http://arxiv.org/abs/2006.16712` (visited on 08/01/2022).

Mohanty, Sharada et al. (Dec. 10, 2020). "Flatland-RL : Multi-Agent Reinforcement Learning on Trains". URL: `https://arxiv.org/abs/2012.05893v2` (visited on 10/02/2021).

Moravčík, Matej et al. (May 5, 2017). "DeepStack: Expert-level Artificial Intelligence in Heads-up No-Limit Poker". In: *Science* 356.6337, pp. 508–513. DOI: `10.1126/science.aam6960`. URL: `https://www.science.org/doi/10.1126/science.aam6960` (visited on 02/15/2022).

Nalepka, Patrick et al. (2021). "Interaction Flexibility in Artificial Agents Teaming with Humans". In: *Proceedings of the Annual Meeting of the Cognitive Science Society* 43.43. URL: `https://escholarship.org/uc/item/9ks6n70q` (visited on 05/27/2022).

Ng, Andrew Y. and Stuart Russell (2000). "Algorithms for Inverse Reinforcement Learning." In: *Icml*. Vol. 1, p. 2.

Nikolaidis, Stefanos et al. (2015). "Efficient Model Learning from Joint-Action Demonstrations for Human-Robot Collaborative Tasks". In: *2015 10th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*. IEEE, pp. 189–196.

Nikolaidis, Stefanos et al. (2016). "Formalizing Human-Robot Mutual Adaptation: A Bounded Memory Model". In: *2016 11th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*. IEEE, pp. 75–82.

OpenAI Five. OpenAI. URL: `https://openai.com/blog/openai-five/` (visited on 08/25/2022).

Osborne, Martin J. and Ariel Rubinstein (1994). *A Course in Game Theory*. The MIT Press. ISBN: 0262150417.

Padakandla, Sindhu (July 13, 2021). "A Survey of Reinforcement Learning Algorithms for Dynamically Varying Environments". In: *ACM Computing Surveys* 54.6, 127:1–127:25. ISSN: 0360-0300. DOI: 10.1145/3459991. URL: https://doi.org/10.1145/3459991 (visited on 09/20/2022).

Padakandla, Sindhu, Prabuchandran K. J., and Shalabh Bhatnagar (Nov. 1, 2020). "Reinforcement Learning Algorithm for Non-Stationary Environments". In: *Applied Intelligence* 50.11, pp. 3590–3606. ISSN: 1573-7497. DOI: 10.1007/s10489-020-01758-5. URL: https://doi.org/10.1007/s10489-020-01758-5 (visited on 09/25/2022).

Palmer, Gregory, Rahul Savani, and Karl Tuyls (May 7, 2019). *Negative Update Intervals in Deep Multi-Agent Reinforcement Learning*. arXiv: 1809.05096 [cs]. URL: http://arxiv.org/abs/1809.05096 (visited on 08/20/2022).

Paquette, Philip et al. (2019). "No-Press Diplomacy: Modeling Multi-Agent Gameplay". In: *Advances in Neural Information Processing Systems*. Vol. 32. Curran Associates, Inc. URL: https://proceedings.neurips.cc/paper/2019/hash/84b20b1f5a0d103f5710bb67a043cd78-Abstract.html (visited on 08/15/2022).

Park, Kwanyoung, Hyunseok Oh, and Youngki Lee (May 3, 2021). *VECA : A Toolkit for Building Virtual Environments to Train and Test Human-like Agents*. DOI: 10.48550/arXiv.2105.00762. arXiv: 2105.00762 [cs]. URL: http://arxiv.org/abs/2105.00762 (visited on 08/11/2022).

Perez-Liebana, Diego et al. (Jan. 23, 2019). *The Multi-Agent Reinforcement Learning in Malm\"O (MARL\"O) Competition*. DOI: 10.48550/arXiv.1901.08129. arXiv: 1901.08129 [cs]. URL: http://arxiv.org/abs/1901.08129 (visited on 08/20/2022).

Pineau, Joelle, Geoffrey Gordon, and Sebastian Thrun (2006). "Anytime Point-Based Approximations for Large POMDPs". In: *Journal of Artificial Intelligence Research* 27, pp. 335–380.

Poundstone, W. (1992). *Prisoner's Dilemma*. Doubleday. ISBN: 978-0-385-41567-5. URL: https://books.google.cz/books?id=9uruAAAAMAAJ.

Press, William H. and Freeman J. Dyson (2012). "Iterated Prisoner's Dilemma Contains Strategies That Dominate Any Evolutionary Opponent". In: *Proceedings of the National Academy of Sciences* 109.26, pp. 10409–10413.

Puig, Xavier et al. (May 3, 2021). "Watch-And-Help: A Challenge for Social Perception and Human-AI Collaboration". URL: http://arxiv.org/abs/2010.09890 (visited on 05/25/2022).

Puterman, Martin L. and Moon Chirl Shin (1978). "Modified Policy Iteration Algorithms for Discounted Markov Decision Problems". In: *Management Science* 24.11, pp. 1127–1137.

Rabinowitz, Neil et al. (July 3, 2018). "Machine Theory of Mind". In: *International Conference on Machine Learning*. PMLR, pp. 4218–4227. URL: `https://proceedings.mlr.press/v80/rabinowitz18a.html` (visited on 10/19/2021).

Redefined, AI et al. (2021). *Cogment: Open Source Framework For Distributed Multi-actor Training, Deployment and Operations*. arXiv: `2106.11345 [cs.AI]`.

Resnick, Cinjon et al. (2018). "Pommerman: A Multi-Agent Playground".

Robinson, David and David Goforth (Jan. 1, 2005). *The Topology of the 2x2 Games: A New Periodic Table*. Vol. 3. Psychology Press. DOI: `10.4324/9780203340271`.

Rousseau, Jean-Jacques (1984). *A discourse on inequality*. Penguin Books Harmondsworth. ISBN: 0140444394.

Rummery, Gavin A. and Mahesan Niranjan (1994). *On-Line Q-learning Using Connectionist Systems*. Vol. 37. Citeseer.

Russell, Stuart J et al. (1995). *Artificial Intelligence A Modern Approach*. Prentice-Hall International. ISBN: 0-13-103805-2.

Sadigh, Dorsa et al. (2016). "Planning for Autonomous Cars That Leverage Effects on Human Actions." In: *Robotics: Science and Systems*. Vol. 2. Ann Arbor, MI, USA, pp. 1–9.

Sartoretti, Guillaume et al. (Feb. 20, 2019). "PRIMAL: Pathfinding via Reinforcement and Imitation Multi-Agent Learning". URL: `http://arxiv.org/abs/1809.03531` (visited on 04/15/2020).

Schaeffer, Jonathan et al. (1992). "A World Championship Caliber Checkers Program". In: *Artificial Intelligence* 53.2-3, pp. 273–289.

Schrittwieser, Julian et al. (Dec. 2020). "Mastering Atari, Go, Chess and Shogi by Planning with a Learned Model". In: *Nature* 588.7839 (7839), pp. 604–609. ISSN: 1476-4687. DOI: `10.1038/s41586-020-03051-4`. URL: `https://www.nature.com/articles/s41586-020-03051-4` (visited on 07/31/2022).

Shih, Andy et al. (Apr. 7, 2021). "On the Critical Role of Conventions in Adaptive Human-AI Collaboration". URL: `https://arxiv.org/abs/2104.02871v1` (visited on 10/12/2021).

Silver, David et al. (2016). "Mastering the Game of Go with Deep Neural Networks and Tree Search". In: *nature* 529.7587, pp. 484–489. DOI: `10.1038/nature16961`.

Silver, David et al. (July 17, 2017). "The Predictron: End-To-End Learning and Planning". In: *Proceedings of the 34th International Conference* *on Machine Learning*. International Conference on Machine Learning. PMLR, pp. 3191–3199. URL: https://proceedings.mlr.press/v70/silver17a.html (visited on 08/26/2022).

Silver, David et al. (Dec. 7, 2018). "A General Reinforcement Learning Algorithm That Masters Chess, Shogi, and Go through Self-Play". In: *Science* 362.6419, pp. 1140–1144. DOI: 10.1126/science.aar6404. URL: https://www.science.org/doi/10.1126/science.aar6404 (visited on 10/17/2022).

Song, Yuhang et al. (Apr. 3, 2020). "Arena: A General Evaluation Platform and Building Toolkit for Multi-Agent Intelligence". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 34.05 (05), pp. 7253–7260. ISSN: 2374-3468. DOI: 10.1609/aaai.v34i05.6216. URL: https://ojs.aaai.org/index.php/AAAI/article/view/6216 (visited on 08/11/2022).

Strouse, DJ et al. (2021). "Collaborating with Humans without Human Data". In: *Advances in Neural Information Processing Systems*. Vol. 34. Curran Associates, Inc., pp. 14502–14515. URL: https://proceedings.neurips.cc/paper/2021/hash/797134c3e42371bb4979a462eb2f042a-Abstract.html (visited on 05/27/2022).

Suarez, Joseph et al. (Mar. 2, 2019). *Neural MMO: A Massively Multiagent Game Environment for Training and Evaluating Intelligent Agents*. DOI: 10.48550/arXiv.1903.00784. arXiv: 1903.00784 [cs, stat]. URL: http://arxiv.org/abs/1903.00784 (visited on 10/17/2022).

Sunehag, Peter et al. (2018). "Value-Decomposition Networks For Cooperative Multi-Agent Learning Based On Team Reward". In: *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*. AAMAS '18. Stockholm, Sweden: International Foundation for Autonomous Agents and Multiagent Systems, 2085–2087.

Sutton, Richard S and Andrew G Barto (2018). *Reinforcement Learning: An Introduction*. MIT press.

Sutton, Richard S., Michael H. Bowling, and Patrick M. Pilarski (Aug. 23, 2022). *The Alberta Plan for AI Research*. DOI: 10.48550/arXiv.2208.11173. arXiv: 2208.11173 [cs]. URL: http://arxiv.org/abs/2208.11173 (visited on 09/30/2022).

Tamar, Aviv et al. (2016). "Value Iteration Networks". In: *Advances in Neural Information Processing Systems*. Vol. 29. Curran Associates, Inc. URL: https://proceedings.neurips.cc/paper/2016/file/c21002f464c5fc5bee3b98ced83963b8-Paper.pdf (visited on 10/17/2022).

Tassa, Yuval et al. (2018). "DeepMind Control Suite". In: *CoRR* abs/1801.00690. arXiv: 1801.00690. URL: http://arxiv.org/abs/1801.00690.

Terry, J. K et al. (2020a). "PettingZoo: Gym for Multi-Agent Reinforcement Learning". In: *arXiv preprint arXiv:2009.14471*.

Terry, Justin K. et al. (2020b). "Agent Environment Cycle Games".

Tesauro, Gerald (2003). "Extending Q-learning to General Adaptive Multi-Agent Systems". In: *Advances in neural information processing systems* 16.

"The Computational Complexity of Agent Design Problems" (2000). In: *Proceedings Fourth International Conference on MultiAgent Systems*, pp. 341–348. DOI: 10.1109/ICMAS.2000.858472. URL: https://ieeexplore.ieee.org/document/858472 (visited on 08/25/2022).

Todorov, Emanuel, Tom Erez, and Yuval Tassa (2012). "MuJoCo: A physics engine for model-based control". In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033. DOI: 10.1109/IROS.2012.6386109.

Tuyls, Karl, Pieter Jan 'T Hoen, and Bram Vanschoenwinkel (Jan. 1, 2006). "An Evolutionary Dynamical Analysis of Multi-Agent Learning in Iterated Games". In: *Autonomous Agents and Multi-Agent Systems* 12.1, pp. 115–153. ISSN: 1573-7454. DOI: 10.1007/s10458-005-3783-9. URL: https://doi.org/10.1007/s10458-005-3783-9 (visited on 08/01/2022).

Tuyls, Karl and Simon Parsons (May 1, 2007). "What Evolutionary Game Theory Tells Us about Multiagent Learning". In: *Artificial Intelligence*. Foundations of Multi-Agent Learning 171.7, pp. 406–416. ISSN: 0004-3702. DOI: 10.1016/j.artint.2007.01.004. URL: https://www.sciencedirect.com/science/article/pii/S0004370207000082 (visited on 08/01/2022).

Vinyals, Oriol et al. (2017). "StarCraft II: A New Challenge for Reinforcement Learning". In: *CoRR* abs/1708.04782. arXiv: 1708.04782. URL: http://arxiv.org/abs/1708.04782 (visited on 10/17/2022).

Vinyals, Oriol et al. (2019). "Grandmaster Level in StarCraft II Using Multi-Agent Reinforcement Learning". In: *Nature* 575.7782, pp. 350–354. DOI: 10.1038/s41586-019-1724-z.

Wang, Rose E. et al. (2020). "Too Many Cooks: Coordinating Multi-agent Collaboration Through Inverse Planning". In: *Proceedings of the 19th International Conference on Autonomous Agents and Multiagent Systems, AAMAS '20, Auckland, New Zealand, May 9-13, 2020*. Ed. by Amal El Fallah Seghrouchni et al. International Foundation for Autonomous Agents and Multiagent Systems, pp. 2032–2034. DOI: 10.5555/3398761.3399065. URL: https://dl.acm.org/doi/10.5555/3398761.3399065 (visited on 10/17/2022).

Witten, Ian H. (1977). "An Adaptive Optimal Controller for Discrete-Time Markov Environments". In: *Inf. Control.* 34.4, pp. 286–295. DOI: 10.1016/S0019-9958(77)90354-0.

Wong, Annie et al. (June 29, 2021). *Multiagent Deep Reinforcement Learning: Challenges and Directions Towards Human-Like Approaches*. DOI: 10.48550/arXiv.2106.15691. arXiv: 2106.15691 [cs]. URL: http://arxiv.org/abs/2106.15691 (visited on 07/14/2022).

Wooldridge, Michael J. (2009). *An Introduction to MultiAgent Systems, Second Edition*. Wiley. ISBN: 978-0-470-51946-2.

Wu, Sarah A. et al. (2021). "Too Many Cooks: Bayesian Inference for Coordinating Multi-Agent Collaboration". In: *Topics in Cognitive Science* 13.2, pp. 414–432. ISSN: 1756-8765. DOI: 10.1111/tops.12525. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/tops.12525 (visited on 04/19/2022).

Wydmuch, Marek, Michał Kempka, and Wojciech Jaśkowski (Sept. 2019). "ViZDoom Competitions: Playing Doom from Pixels". In: *IEEE Transactions on Games* 11.3, pp. 248–259. ISSN: 2475-1502, 2475-1510. DOI: 10.1109/TG.2018.2877047. arXiv: 1809.03470 [cs, stat]. URL: http://arxiv.org/abs/1809.03470 (visited on 09/30/2022).

Zheng, Yan et al. (2018). "A Deep Bayesian Policy Reuse Approach Against Non-Stationary Agents". In: *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*. Ed. by Samy Bengio et al., pp. 962–972. URL: https://proceedings.neurips.cc/paper/2018/hash/85422afb467e9456013a2a51d4dff702-Abstract.html (visited on 10/17/2022).

Zhu, Hao, Graham Neubig, and Yonatan Bisk (2021). "Few-Shot Language Coordination by Modeling Theory of Mind". In: *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*. Ed. by Marina Meila and Tong Zhang. Vol. 139. Proceedings of Machine Learning Research. PMLR, pp. 12901–12911. URL: http://proceedings.mlr.press/v139/zhu21d.html (visited on 10/17/2022).

Ziebart, Brian D. et al. (2009). "Planning-based prediction for pedestrians". In: *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3931–3936. DOI: 10.1109/IROS.2009.5354147.