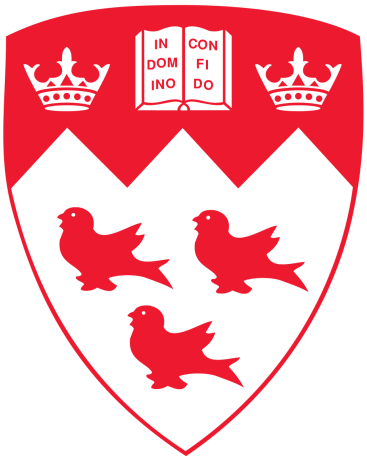
Coordination in Cooperative Multi-Agent Learning

Paul Barde

Department of Electrical & Computer Engineering
McGill University
Montréal, Québec, Canada



December 13, 2023

A thesis submitted to McGill University in partial fulfillment of the requirements of the degree of Doctorate of Electrical & Computer Engineering

Image generated by Midjourney from the prompt “One-line drawing of climbers helping each other to reach a summit”.

Contents

Abstract

Exploring the process by which autonomous agents coordinate represents a pivotal advancement toward emulating populations, which encompasses diverse applications in robotics, game theory, economics, and social sciences. Specifically, the mechanisms that allow cooperating learners to converge toward coherent team strategies remain poorly understood, and yet, understudied. The myriad of challenges that hinder concurrent learning – environment non-stationarity, intricate credit assignments, exponential complexity, etc. – has fragmented the focus of the literature such that the underlying mechanisms of successful multi-agent coordination remain to be pinpointed. This work puts forward and explores three key elements of coordination, namely, shared incentives, interactions, and the use of internal models. We scrutinize this take by inspecting how these components can enable coordination in several multi-agent learning paradigms that cover the well-established Multi-Agent Reinforcement Learning framework, its offline interaction-less variation, and the Architect-Builder Problem, a novel reward-less interactive learning setting. Our research derives fresh insights into fostering coordination and while the implications of multi-agent learning extend across various fields, we are particularly interested in planning solutions to societal and ecological challenges by simulating how populations would react to changes in their environment.

Résumé

L'exploration du processus par lequel les agents autonomes se coordonnent représente un progrès décisif vers l'émulation des populations, qui englobe diverses applications en robotique, en théorie des jeux, en économie et en sciences sociales. Plus précisément, les mécanismes qui permettent à des apprenants coopérants de converger vers des stratégies d'équipe cohérentes restent mal compris et encore peu étudiés. La myriade de défis qui entravent l'apprentissage simultané - non-stationnarité de l'environnement, partage des récompenses, complexité exponentielle, etc. – a fragmenté la littérature de telle sorte que les mécanismes sous-jacents d'une coordination multi-agents réussie n'ont pas encore été mis en évidence. Ce travail met en avant et explore trois éléments clés de la coordination, à savoir les incitations partagées, les interactions et l'utilisation de modèles internes. Nous examinons comment ces éléments peuvent permettre la coordination dans plusieurs paradigmes d'apprentissage multi-agents qui couvrent le cadre bien établi de l'apprentissage par enforcement multi-agent, sa variation hors ligne sans interaction, et le problème de l'architecte et du constructeur, un nouveau cadre d'apprentissage interactif sans récompense. Notre travail apporte de nouvelles perspectives pour favoriser la coordination et, bien que les implications de l'apprentissage multi-agents s'étendent à divers domaines, nous sommes particulièrement intéressés par la recherche de solutions aux défis sociétaux et écologiques par la simulation des réactions d'une population aux modifications de son environnement.

Acknowledgements

For their supervision, support, inspiration, advice, guidance, and resources, I would like to thank:

- Derek Nowrouzezahrai and Christopher Pal.
- Tristan Karch, Julien Roy and Wonseok Jeon.
- Emmanuel Rachelson, Wulfram Gerstner, Auke Ijspeert and Nicolas Mansard.
- Marc G. Bellemare, Cédric Colas, Olivier Delalleau, Jakob Foerster, Félix G. Harvey, Hengyuan Hu, Erwan Lecapentier, David Meger, Clement Moulin-Frier, Pierre-Yves Oudeyer, Joelle Pineau, Luis Pineda, Olivier Pomarez, Eugene Vinitsky, Amy Zhang.
- Mila and LaForge ecosystems.
- Compute Canada.
- Midjourney for generating the cover image.
- Slidesgo and Flaticon for visuals.

Contribution to Original Knowledge

Chapter 4:

- noting the counter-intuitive observation that current Centralized Training and Decentralized Execution (CTDE) algorithms do not outperform their decentralized counterparts at discovering optimal team strategies with shared rewards on several coordination environments.
- coordination-intensive tasks in multi-agent particle environment.
- definition of coordination in terms of predictability, synchronicity, and coherence.
- analysis of the benefits of coordination on learning in a novel well-controlled environment and task.
- design of incentives for coordination according to the proposed definition.
- practical implementation of these incentives for Multi-Agent Reinforcement Learning algorithms.
- detailed learning and coordination analysis of these methods.
- rigorous hyperparameter tuning and experiment design for fair and significant comparison.

Chapter 5:

- identifying the Offline Coordination Problem (OCP) and decomposing it into Strategy Agreement (SA) and Strategy Fine-Tuning (SFT).
- noting that current state-of-the-art methods fail at offline coordination, even with online fine-tuning.
- noting that even centralized methods fail at Strategy Fine-Tuning.
- new datasets and partially observable tasks in multi-agent MuJoCo.
- use of world models for multi-agent offline coordination.
- first model-based offline multi-agent algorithm.
- method that solves both Strategy Agreement and Strategy Fine-Tuning.
- devised new practical methods to use world models: adaptive rollouts based on ensemble uncertainty and clipping to the dataset's bounding box.

Chapter 6:

- novel interactive learning setting: the Architect-Builder Problem (ABP).
- tasks and Architect-Builder BuildWorld environment.
- use of high-level shared intent for cooperation.
- use and implementation of interaction frames for multi-agent learning.
- formulation of shared intent as guiding with multi-step Monte Carlo Tree Search (MCTS) and self-imitation.
- Architect-Builder Iterated Guiding (ABIG) algorithm.
- analysis of the learning mechanisms and dynamics for the emergence of communication.

Chapter 7:

- proofs for Adversarial Imitation Learning (AIL) optimal decomposition in trajectory and transition space.
- analytical optimal generator policy for a given discriminator.
- can directly retrieve the optimal generator policy in practice.
- removing the need for policy training in AIL.
- corresponding Imitation Learning (IL) algorithms from both trajectories and transitions.
- observation of the importance of Gradient Penalty (GP) in Generative Adversarial Imitation Learning (GAIL).
- tasks and corresponding datasets in a modified version of the Pommerman environment.
- imitation datasets for classic control, Box2D, and MuJoCo environments.
- granular performance imitation dataset in MuJoCo.

Publications and Contribution of Authors

1. Chapter 4 features the work that we published in Roy et al. (2020). Julien started the project with an initial draft on multi-agent coordination and predictability. He also started the codebase with the environments and the MADDPG baseline. At that point, Paul embarked on the project. Together they simplified and fixed the codebase so as to have a working point for the baselines, they also implemented and designed the experimental procedure of the hyperparameter selection and retraining together. Paul and Julien designed TeamReg and CoachReg together. Paul implemented and ran the experiments and analysis for CoachReg while Julien did it for TeamReg. Paul designed and implemented the toy experiment analysis on coordination. Paul implemented the parameter-sharing baseline. Julien interfaced the code with the Football environment while Paul ran the corresponding experiments. Both contributed to the entirety of the writing of the paper; Julien focused more on the background, environments, and content related to TeamReg, while Paul focused on the sections related to CoachReg, motivation, and experimental design.
2. Chapter 5 feature the work that we submitted to NeurIPS 2023 main conference (Barde et al., 2023). As the sole first author of this work, Paul carried the project from its definition to the implementation of the codebase and the writing of the paper. Co-authors contributed through bi-weekly meetings focused on giving updates on the project and discussing problems and next steps.
3. Chapter 6 features the work that we published in Barde et al. (2022). Paul started the project with the idea of using IRL for cooperation and coordination in the absence of rewards. Paul and Tristan designed together the Architect-Builder problem setting. Paul derived the theoretical formalism for it and implemented the method, the environment, and the tasks. Paul ran the experiments on learning, transfer, and analysis in the BuildWorld while Tristan ran the experiments on the learning mechanisms in the toy problem. Both Paul and Tristan contributed to the entirety of the writing of the paper; Paul focused on the methods, the background, the related works, and the results while Tristan focused on the problem definition and the learning mechanisms.

4. Chapter 7 features the work that we published in Barde et al. (2020). Paul started the project, derived the methods, and wrote a first draft. Paul implemented the methods, the baselines, and the environments as well as collected the demonstrations. Paul derived the proofs and ran the experiments for the classic control and Box2D tasks, Julien ran the experiments in Pommerman, and Wonseok ran the experiments on MuJoCo. All three contributed to the whole writing of the paper, Paul focused on the background, methods, and related works; Julien focused on the Pommerman results, and Wonseok on the MuJoCo results.

Acronyms

ABIG	Architect-Builder Iterated Guiding.
ABP	Architect-Builder Problem.
AI	Artificial Intelligence.
AIL	Adversarial Imitation Learning.
AIRL	Adversarial Inverse Reinforcement Learning.
ASAF	Adversarial Soft Advantage Fitting.
ASQF	Adversarial Soft Q Fitting.
AWR	Advantage Weighted Regression.
BC	Behavioral Cloning.
BCI	Brain-Computer Interface.
CBR	Case-Based Reasoning.
CCM	Convergent Cross Mapping.
CEA	Coevolutionary Algorithms.
CMAL	Cooperative Multi-Agent Learning.
CoCo	Co-Construction.
COMBO	Conservative Offline Model-Based policy Optimization.
CQL	Conservative Q-Learning.
CTDE	Centralized Training and Decentralized Execution.
CUD	Categorical Uniform Distribution.
D4RL	Datasets for Deep Data-Driven Reinforcement Learning.
DAI	Distributed Artificial Intelligence.
DDPG	Deep Deterministic Policy Gradient.
Dec-MDP	Decentralized Markov Decision Process.
Dec-POMDP	Decentralized Partially Observable Markov Decision Process.
DM	Distribution Matching.
DPS	Distributed Problem Solving.

EC	Evolutionary Computation.
FO	Full Observability.
GAE	Generalized Advantage Estimator.
GAIL	Generative Adversarial Imitation Learning.
GAN	Generative Adversarial Network.
GP	Gradient Penalty.
GT	Game Theory.
HRI	Human-Robot Interaction.
IBC	Independent Behavioral Cloning.
ICQ	Implicit Constraint Q-learning.
ICQL	Independent Conservative Q-Learning.
IL	Imitation Learning.
IOMAR	Independent Offline Multi-agent reinforcement learning with Actor Rectification.
IQL	Implicit Q-Learning.
IRL	Inverse Reinforcement Learning.
ITD3	Independent Twin Delayed Deep Deterministic Policy Gradient.
JS	Jensen-Shannon.
KL	Kullback–Leibler.
LOLA	Learning with Opponent Learning Awareness.
MABCQ	Multi-Agent Batch Constrained Q-learning.
MADDPG	Multi-Agent Deep Deterministic Policy Gradient.
MAIQL	Multi-Agent Implicit Q-Learning.
MAMuJoCo	Multi-Agent MuJoCo.
MAPPO	Multi-Agent Proximal Policy Optimization.
MARL	Multi-Agent Reinforcement Learning.
MAS	Multi-Agent System.
MaxEnt	Maximum Entropy.
MCTS	Monte Carlo Tree Search.
MDP	Markov Decision Process.
MG	Markov Game.
ML	Machine Learning.

MLP	Multi Layer Perceptron.
MOMA-PPO	Model-based Offline Multi-Agent Proximal Policy Optimization.
MOPO	Model-based Offline Policy Optimization.
MOReL	Model-based Offline Reinforcement Learning.
MSE	Mean Squared Error.
OCP	Offline Coordination Problem.
OMAR	Offline Multi-agent reinforcement learning with Actor Rectification.
PG	Policy Gradient.
PGM	Probabilistic Graphical Model.
PO	Partial Observability.
POMDP	Partially Observable Markov Decision Process.
PPO	Proximal Policy Optimization.
ReLU	Rectified Linear Unit.
RG	Repeated Game.
RL	Reinforcement Learning.
ROMI	Reverse Offline Model-based Imagination.
SA	Strategy Agreement.
SAC	Soft Actor-Critic.
SARSA	State-Action-Reward-State-Action.
SEM	Standard Error of the Mean.
SFT	Strategy Fine-Tuning.
SQIL	Soft Q Imitation Learning.
TD3	Twin Delayed Deep Deterministic Policy Gradient.
ToM	Theory of Mind.
TRPO	Trust-Region Policy Optimization.
UCB	Upper Confidence Bound.
UCT	Upper Confidence bound applied to Trees.
UED	Unsupervized Environment Design.
ZSC	Zero-Shot Coordination.

Chapter 1

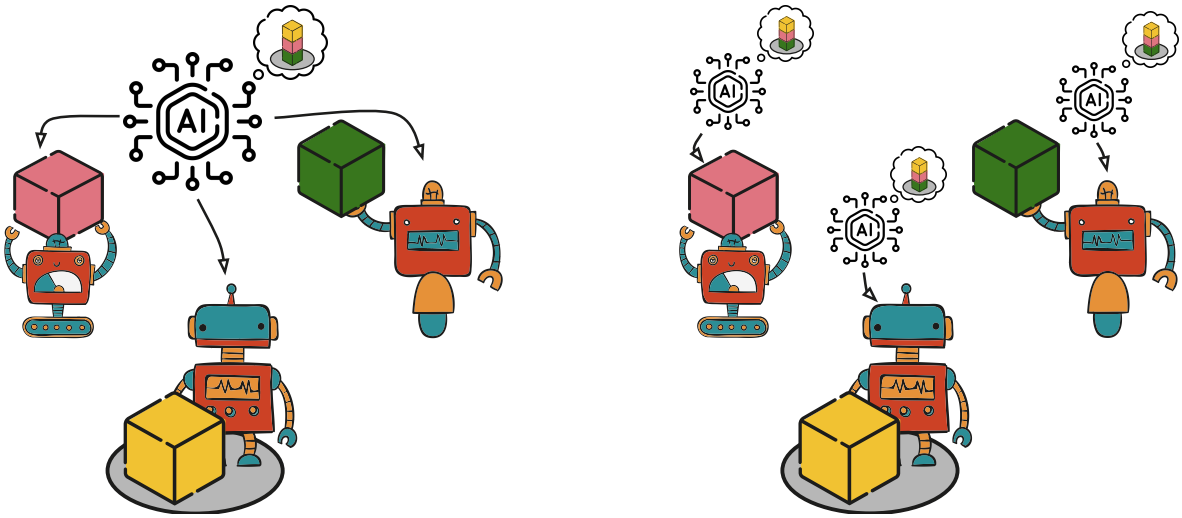
Introduction

As a society, we face pressing challenges such as increasing unemployment rates, wealth inequality, biodiversity loss, and climate change. These complex issues are inherently multi-agent and require individuals and groups to adapt to changes in their environment (Börgers and Krahmer, 2015). In this thesis, we take a modest first step towards addressing these societal and environmental struggles by investigating Cooperative Multi-Agent Learning (CMAL) and exploring how autonomous agents learn to cooperate and coordinate. Through this research, we hope to contribute to a deeper understanding of multi-agent behavior and pave the way for new solutions to the pressing challenges we face today.

We propose to consider the collective emergence of complexity and motivate this approach by taking inspiration from human evolution. Many argue that the concept of socio-cultural cognitive niche played a major role in the successful evolution of humans and the rise of societies (Boyd et al., 2011; Fuentes, 2017). This theory poses that the development of individuals – and of their cognitive skills in particular – is deeply intertwined with the cultural practices and social interactions present in their environment. Humans evolve to create and transmit cultural knowledge and practices, which in turn shape their environment and create new selection pressures. Those selection pressures favor some cultures and practices over others, which lay the conditions for the Baldwin effect to operate. The Baldwin effect postulates that traits that are initially learned through experience can become genetically encoded over time if they provide a selective advantage. Indeed, natural selection will favor individuals with a genetic predisposition for the learned trait (Simpson, 1953). Reciprocally, the natural selection of culturally promoted traits at the scale of individuals impacts the development of societies and affects what is culturally encouraged. Arguably, the cognitive niche and the Baldwin effect provided a rich and dynamic environment in which individual experience could shape the course of evolution. For some, this explains the emergence of cooperation, sociality, tools, technologies, educational practices, social norms, language, and so on. Crucially, it also describes how these attributes mutually shaped the evolution of societies and individuals, leading the way to our very particular cognition (Sterelny, 2003;

Sweller, 2003; Boyd and Richerson, 2009; Pinker, 2010; Hayes and Sanford, 2014; Boyd, 2018). The cognitive niche argument suggests that, in order to understand human cognition – and maybe eventually try to emulate it – one should study both the cognitive processes of individuals and the social interactions and cultural context that give rise to it (Galantucci and Garrod, 2011). We embrace this idea and focus on exploring the mechanisms that allow multiple agents to learn, cooperate, socialize, and, maybe, eventually co-evolve into something as complex as humans and societies.

Since our final aim is to simulate and optimize behaviors, we believe that it is most relevant to tackle this problem directly *in-silico*. Consequently, we take an Artificial Intelligence (AI)-centered approach rather that a more biological or anthropological one.



(a) DPS and *team learning*. Centralized.

(b) MAS and *concurrent learning*. Decentralized.

Figure 1.1: Illustration of the two approaches to Distributed Artificial Intelligence (DAI) and their respective learning paradigms. The cooperative task consists in piling boxes in a certain order at the location specified in grey. Agents must coordinate if they are to succeed: first, the middle robot has to grab the yellow box and make room so that the robot on the right side can drop the green box on the target. Then, the robot on the left side must place the pink box on top of the green one, and finally, the middle robot can complete the task by piling the yellow box. (a) Distributed Problem Solving (DPS) and *team learning*: a single centralized algorithm controls all the agents. (b) Multi-Agent System (MAS) and *concurrent learning*: each agent is its own autonomous individual learner. This work focuses on MAS and concurrent learners, a setting referred to as Cooperative Multi-Agent Learning (CMAL).

Distributed Artificial Intelligence (DAI) improves the performance, scalability, and ro-

bustness of AI systems by relying on multiple distributed artificial agents (Ferber and Weiss, 1999). DAI can be split into two sub-fields. On the one hand, Distributed Problem Solving (DPS) – see Figure 1.1 (a) – requires centralized control in order to construct a collective solution by decomposing the problem and distributing simple sub-tasks to executive agents; on the other hand, Multi-Agent System (MAS) – see Figure 1.1 (b) – refers to autonomous self-governed agents jointly interacting (Panait and Luke, 2005). In short, DPS agents are assumed *benevolent* and *cooperative* by default, while MAS agents are *selfish* in essence and care about their own goals (Nwana et al., 1997; Mohamed and Huhns, 2001). Agents with aligned – possibly shared – or conflicting – possibly opposing – goals respectively induce *cooperative* or *competitive* situations. We care about MAS and cooperation.

In the broad field of AI, we are interested in Machine Learning (ML) and we specifically focus on Reinforcement Learning (RL). RL is an area of ML that is best used to learn in sequential decision-making problems. RL agents are autonomous agents that gather experiences by interacting with their environment and use these experiences to learn and improve their behavior towards achieving their goals. The key concepts of sequential decision-making are actions, states, rewards, transition functions – or environment dynamics –, and policies. Interactions unfold with the agent observing the current state and using its policy to select the most appropriate action. This action affects the environment that transitions to the next state. The agent receives a reward that rates how "good" is the transition with respect to the goal (see Figure 1.2). A sequence of states, actions, and rewards triplets is called a trajectory and the agent aims at maximizing the rewards it can collect along those. The policy encodes the agent's behavior, it defines the action it ought to take given the current state. Most RL learning algorithms have the agent estimate values (or state-values) that capture the outcome of following the current policy from the current state, and Q-values (or state-action values) that predict the outcome of taking a given action and then following the current policy from the current state. In a nutshell, RL agents strive to discover and reinforce the behaviors that yield the most rewards through trial-and-error interactions with the environment.

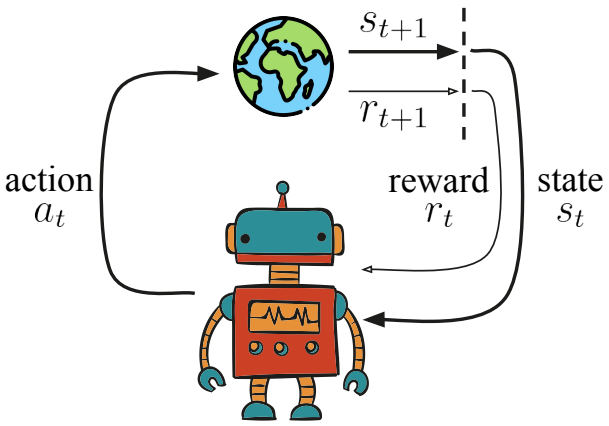


Figure 1.2: Sequential decision making: the agent observes the current state, takes an action to make the environment transition to the next state, and receives a reward for it.

Our work aims at training autonomous learners that adapt in response to others so as to analyze how cooperative agents interact and give rise to emergent group strategies. This setting is referred to as Cooperative Multi-Agent Learning (CMAL) (Panait and Luke,

2005). In CMAL cooperation is not governed but must be learned and enforced by the agents themselves: it is often the result of *coordination*. Coordination is the property of an intelligent agency that ensures that a group of agents interact coherently as a single unit (Weiss, 1999; Sycara, 1989). Coherence in this context means that agents produce actions that blend without interfering, avoiding deadlocks and livelocks while they solve the task. See Figure 1.1 for an example of a cooperative task requiring coordination.

The decentralized nature of CMAL – where entities only have local views, goals, and knowledge – is likely to degenerate into a chaos of conflicting agents if they fail to coordinate. This may prevent the group from achieving their task, eventually defeating the purpose of doing decentralized control in the first place. Specifically, coordination is required in order to (Durfee et al., 1989; Nwana, 1996; Jennings, 1996):

1. be exhaustive and ensure that all the essential elements of the overall problem are being tackled by at least one agent,
2. meet global constraints: if acceptable solutions require the team to satisfy budgets or time limits, agents must account for the activity and resource consumption of other agents in order to meet these requirements,
3. leverage the expertise, resources, and information that is distributed across different agents. It is more than likely that no one agent can solve the overall problem on its own, and
4. to account for the dependencies between agents' behaviors, either due to interdependent (sub-)goals or because they are interacting with the same environment.

Note that cooperation does not necessarily require coordination as incoherent behavior of some agents might still enable the team to succeed as a whole.

In Figure 1.3 (a) where two agents must stack boxes, one agent might act at random and not contribute to solving the task, waiting for the other agent to complete the pile. Reciprocally, coordination might occur without strict cooperation, such as in competitive or partially aligned situations. Or when agents do not realize that their goals are aligned because they have different abilities, knowledge, or beliefs. In such situations, coordination is often referred to as *negotiation* to highlight the win-win trade-off that agents seek (Weiss, 1999). Agents with orthogonal goals (i.e., neither aligned nor conflicting) might also end up coordinating, Figure 1.3 (b) shows for example an agent getting out of the way of another agent running towards it.

Interestingly, even in situations where coordination is not required and agents could act in isolation, it is often more efficient for them to coordinate. For instance, the information discovered by one agent can be shared with other agents, enabling them to solve their problems faster (Clearwater et al., 1991). In a scenario where each agent is looking for a different object, if they all communicate the location of the objects they encounter during the search, then, overall, agents are likely to find their own object faster.

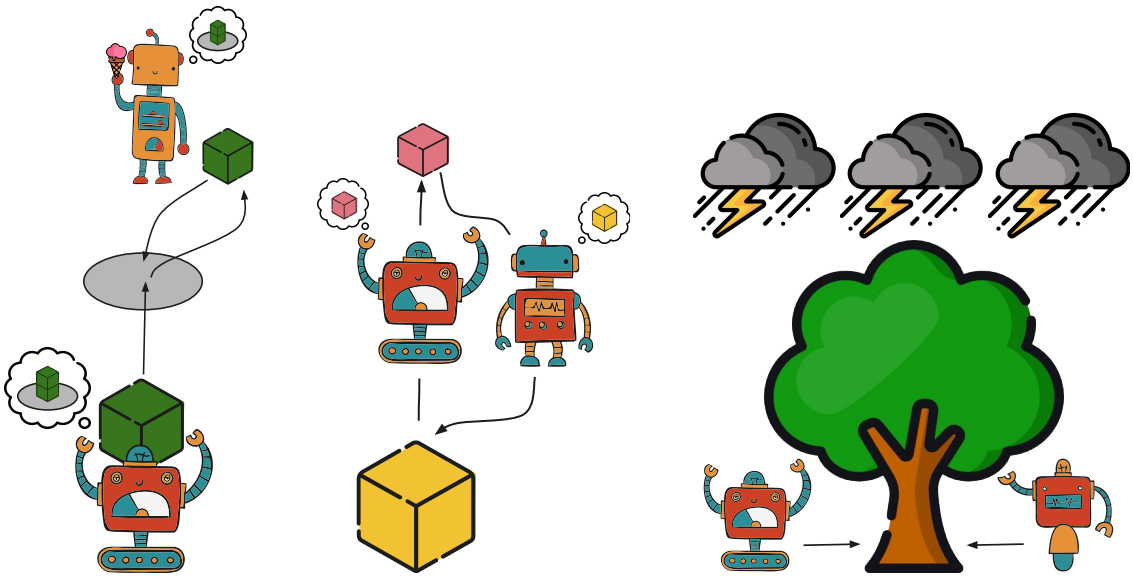


Figure 1.3: Illustration of the difficulty to assess if agents coordinate by examining it from and external perspective. (a) Uncoordinated teams can sometimes succeed in cooperative tasks: the agent in the back doesn't participate in solving the task. (b) Agents can coordinate even if they do not cooperate, here the agent on the right coordinates with the other agent in order to dodge it while going for its target. (c) Agents unaware of each other can appear coordinated due to external factors, here two robots synchronously rush towards the same tree in order to take shelter.

Conversely, coherent actions might arise without coordination. Figure 1.3 (c) illustrates this: envision people sitting in a park, that, as the result of a sudden downpour, start running towards a tree in the middle of the park because it is the only source of shelter (Searle, 2002). Each person aims at avoiding becoming wet and while they are aware of what others are doing and what might be their goals, it does not affect their actions: it is uncoordinated behavior.

In light of this, Jennings (1996) pointed out a key aspect of coordination: it is in general impossible to determine whether or not agents have coordinated their actions when looking at their behaviors from an external perspective. Indeed, agents can try to coordinate but fail to do so and yield incoherent strategies if they lack computational power or rely on incorrect predictions of each other's demeanors and environmental dynamics. This incapacity in assessing coordination by looking at behavior alone is why Jennings (1996) proposes to investigate the internal mechanisms that drive agents' interactions, namely their beliefs, knowledge, preferences, intentions, and so on. Here, we follow this idea and focus on the role of *interactions*, *internal models* of others and of the world, and *shared incentives*.

In CMAL, agents are concurrent learners that interact with their peers and their environment in order to reach their goals. *Interactions* are central to this learning process as they are both the end – agents collectively interacting in an efficient strategy – and the mean – it is through interactions that the learners size the world around them and find the best way to adapt to it. While interactions are also at the core of single agent learning, they are in that case limited to interactions with the environment (Beer, 1995). On the other hand, with multiple agents, not only are environment interactions more intricate – as multiple agents are simultaneously experiencing the world – but also a new type of interaction appears: inter-agent interactions. Since the agents are concurrently learning and changing, these interactions can be highly dynamic and vary over time, making it challenging for agents to interpret and leverage them.

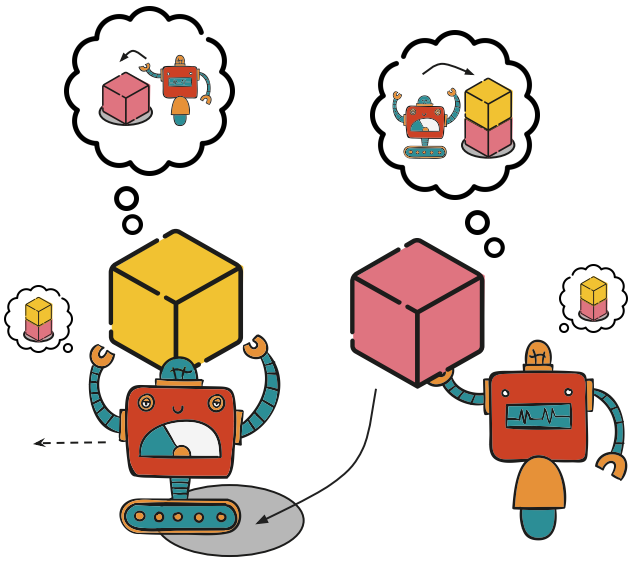


Figure 1.4: Illustration of two agents using their internal model to predict each other's behavior and successfully coordinate their interactions. They share the same objective of piling boxes in a specific order at the grey location. The left-side agent must move left and make room for the right-side agent to place the pink box, then the left-side agent can complete the task by stacking the yellow box on top of the pink box.

Agents must reason about other agents' behaviors. Communication facilitates this by allowing agents to inform others about their intentions, goals, and beliefs. Yet, agents can also achieve this without communicating provided that they possess internal models of each other's behaviors (Huhns and Singh, 1994). Theory of Mind (ToM) refers to the ability of an individual to attribute mental states to himself and others. By reasoning about mental states such as intention, knowledge or belief, one can infer the behavior of others (Premack and Woodruff, 1978) and fruitfully interact with them to coordinate. To be more efficient in terms of interactions, individual agents might also build internal world models so as to estimate how the different team strategies would perform. They can then pick the most promising one. Finally, *shared incentives*, such as aligned goals, conventions, and protocols play a key role in the ability of a group to coordinate (Jennings, 1996; Weiss, 1999; Cao et al., 2018). Figure 1.4 illustrates two agents that successfully coordinate their interactions and achieve the objective they share by predicting each other's behavior.

The following Related Works Chapter 2 discusses the existing literature on coordination in CMAL and how this challenge has been previously tackled, highlighting the role of interactions, internal models, and shared incentives. Then, the Background Chapter 3 pro-

vides the necessary theoretical tools to understand the methods described in this document. Chapters 4, 5, 6 and 7, present work on coordinating concurrent learners and emphasize the importance of interactions, internal models, and shared incentives in different settings and under different assumptions. Chapter 4 investigates cooperative Multi-Agent Reinforcement Learning (MARL) and shows how, even when agents share the same reward function, coordination and team performance can be improved by enforcing additional shared incentives – social norms that agents strive to follow – such as predictability or synchronicity. Chapter 5 examines the Offline Coordination Problem (OCP) that arises in Offline MARL where agents cannot directly interact with one another to coordinate. There, we propose a solution based on building an internal world model that can be used to simulate agents' interactions. Chapter 6 questions a common, yet restrictive, assumption in CMAL: all the agents are aware of the objective they must complete in the sense that they all receive an external reward. Most of the time this reward is a group reward that is the same – i.e. shared – across agents. We define the novel learning setting arising from removing such a strong assumption and present a method to solve it: agents coordinate by evolving a communication protocol, solving the task only through interactions by modeling the other agents, and assuming a high-level shared intent. Chapter 7 discusses the importance of Theory of Mind (ToM) and building inner models of other agents' behaviors. Noting that current methods are rather complex and computationally intensive, it proposes a simpler, yet competitive and more robust method. Finally, the last chapter synthesizes the totality of our findings, highlighting limitations and proposing future directions.

Chapter 2

Related works

We review seminal and more contemporary works on Coordination in Cooperative Multi-Agent Learning (CMAL). After a thorough survey on CMAL, the focus shifts towards coordination. First, we cover the classical approach to coordination in Multi-Agent System that consider non-learning procedural agents. Then we examine the setting that is most interesting to us and that deals with concurrent learners striving to coordinate. Notably, we highlight the importance of interactions, internal models, and shared incentives.

2.1 Cooperative Multi-Agent Learning

The inherent complexity and diversity of multi-agent solutions make most supervised learning methods impractical as it becomes intractable to provide data points for all the possible situations that could arise. Consequently, formulations of the Cooperative Multi-Agent Learning problem are often reward-based (Jennings, 1996). Notable exceptions of supervised multi-agent learners include (Gmytrasiewicz, 1992; Williams, 2004; Garland and Alterman, 2004).

Reward-based formulations are mainly tackled with either Reinforcement Learning (RL) (Sutton and Barto, 2018) and Evolutionary Computation (EC) algorithms (Back, 1996), or to be more specific, with their multi-agent extensions Multi-Agent Reinforcement Learning (MARL) (Littman, 1994) and Coevolutionary Algorithms (CEA) (Wiegand, 2004). Unfortunately, the dynamic nature of learning in Distributed Artificial Intelligence breaks most of the assumptions required to derive the theoretical guarantees of these methods.

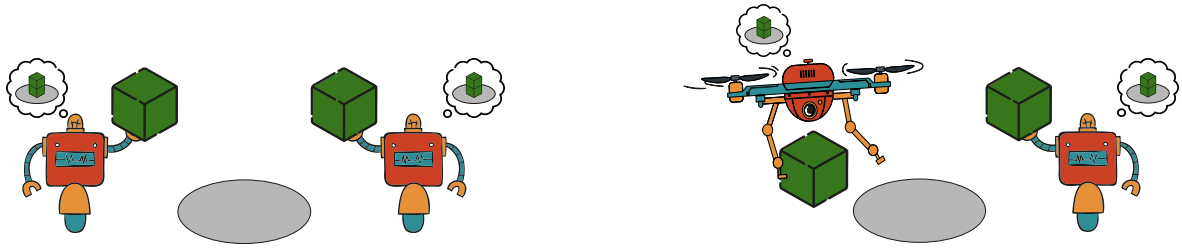
Most approaches to CMAL can be divided into *team learning* – that covers DPS, i.e., a single learner derives the behavior for the entire group of agents– and *concurrent learning* – to tackle MAS where multiple learners concurrently interact while searching for their optimal behavior (Jennings, 1996). Please see Figure 1.1 for an illustration of these two learning paradigms.

2.1.1 Team Learning

Team learning (Figure 1.1 (a)) refers to a single learner deriving the optimal behavior of all the agents in the team simultaneously. This approach is closest to traditional machine learning and one can often use "off-the-shelf" methods. Yet, the algorithms explore the set of strategies of a whole group of agents instead of just one single agent. Thus, even if team learning removes the game-theoretic aspect of concurrent learning, it must address the *emergent complexity* challenge of MAS (Walker and Wooldridge, 1995; Van Dyke Parunak and Brueckner, 2001; Dessalles et al., 2007; Teo et al., 2013):

1. the set of joint behaviors is exponential in the number of agents, and
2. interactions of simple individual behaviors can build up into very complex and unpredictable team patterns.

Additionally, team learning requires centralization, meaning that the learning algorithm must be able to access all the resources which can be unpractical if compute and/or data are distributed. Most research in team learning has focused on *representing* the different candidate solutions that the learner is deriving – i.e., the different team strategies – in relationship with the degree of heterogeneity among the controlled agents. A *homogeneous learner* learns a unique agent behavior and applies it to all the members of the team. In contrast, a *heterogeneous learner* specifies a different behavior for each member. Note that heterogeneous qualifies the behaviors and not the agents directly, for instance, a heterogeneous team could be composed of identical robots learning with identical algorithms but that have discovered different strategies. Obviously, teams composed of different robots with different capabilities (available actions and sensors) must be heterogeneous as the same behavior cannot be applied to all the agents. See Figure 2.1 for an illustration.



(a) A homogeneous team. (b) A heterogeneous team.

Figure 2.1: Illustration of homogeneous vs. heterogeneous teams.

Heterogeneous teams promise more efficient teams of specialized agents at the cost of larger search spaces and memory consumption (Good, 2000). Most research is then interested in determining if specialist agents are required to solve a given problem. On the one hand, homogeneous agents are well suited for tasks in which a single agent can perform well (for

example foraging) regardless of the difficulty of the task (Balch, 1998; Potter et al., 2001). On the other hand, heterogeneity promotes robustness and redundancy while being able to leverage specialized agents in tasks that can be decomposed into sub-tasks (Bongard, 2000). A trade-off, often referred to as *hybrid learners*, can be achieved by dividing the team into squads of homogeneous agents (Luke et al., 1998a,b). Hara (1999) proposes an automatic grouping technique to tackle the compromise between specialization gains and increased search-space size that arises with hybrid teams. The most popular problem domains being tackled with team learning are predator-prey environments (Haynes et al., 1995b,a, 1996; Haynes and Sen, 1996; Haynes et al., 1997) and RoboCup Simulator (Luke et al., 1998a,b; Andre and Teller, 1999).

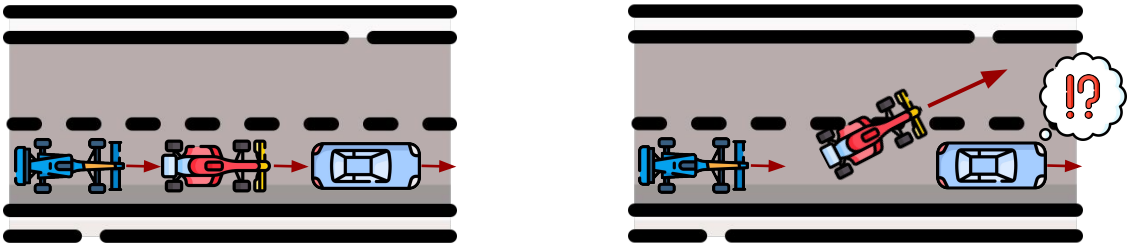
Finally, a line of work investigates which machine learning approach should be used between RL and EC (Salustowicz et al., 1997; Sałustowicz et al., 1998). Model-based variations on Q-learning (Watkins and Dayan, 1992), such as prioritized sweeping (Moore and Atkeson, 1993), appear to be the most promising avenue (Wiering et al., 1999).

2.1.2 Concurrent Learning

An alternative to team learning is concurrent learning (Figure 1.1 (b)) where separate learning processes deal with different aspects of the tasks. In practice in MAS, one learning process is assigned to each agent. This makes concurrent learning suited to problems that can, and should, be decomposed (Jansen and Wiegand, 2003). Indeed, when decomposing the problem, the initial joint-behavior search space is decomposed into smaller, separate individual-behavior spaces. Provided that the individual agent behaviors are not too strongly coupled to each other, this decomposition can result in drastic reductions in terms of search time and complexity. This also allows for more flexibility in distributing resources, such as data and computing power, across learning processes. Yet, while most work points toward concurrent learning outperforming team learning (Bull and Fogarty, 1994; Iba, 1996, 1998), there exist domains that remain best tackled with team learning (Miconi, 2003).

The core challenge for concurrent learners is that their environment – i.e., the other concurrent learners – is co-adapting to them. This breaks most machine learning assumptions by making obsolete the context to which a learner adapted its behavior and can potentially completely ruin a learner's improvement. This is often referred to as *non-stationarity* or *co-adaptation* in concurrent multi-agent learning (Sandholm and Crites, 1996; Weinberg and Rosenschein, 2004). Indeed, an agent's environment is composed of other agents and because these other agents are learning, their behavior is changing over time. Thus, through the eyes of an agent, the environment it is interacting with is changing over time, i.e, it is non-stationary. The simplest approach to the non-stationarity problem is to ignore other agents and consider that they are part of a dynamical environment to which our learner must adapt (Schmidhuber, 1996; Zhao and Schmidhuber, 1996; Schmidhuber and Zhao, 2005). Yet this assumption ignores a crucial aspect of concurrent learning which is that a learner's envi-

ronment – i.e., the other agents – is not just dynamical but actively *adapting* to the learner. See Figure 2.2 for an illustration. A more sophisticated way to address the non-stationarity problem is to adopt the Centralized Training and Decentralized Execution (CTDE) paradigm (Oliehoek et al., 2008; Kraemer and Banerjee, 2016). In Multi-Agent Reinforcement Learning, CTDE algorithms train centralized critics to approximate the joint value functions and use them to optimize decentralized actors that can be deployed autonomously at test time (Lowe et al., 2017; Foerster et al., 2018b; Rashid et al., 2018; Iqbal and Sha, 2019; Foerster et al., 2019).



(a) Initially learned policy. (b) Other agents adaptation.

Figure 2.2: Illustration of concurrent learning-induced non-stationarity: Initially (a) the front car learns to regulate traffic by slowing down which makes other agents also reduce their speed. Yet, the other agent eventually learns to overtake and bypasses the agent's action making this latter's behavior and experiences obsolete. Illustration inspired by Liang and Liaw (2018).

Work in concurrent learning breaks down into three main areas:

(a) *the credit assignment* problem that deals with figuring out which agent in the team to hold accountable for the team reward in order to appropriately reinforce or discourage the corresponding behaviors,
(b) *the dynamics of learning* related to concurrent learners and how they coordinate despite the induced non-stationarity, and
(c) *the modeling of other agents* so as to fine-tune one's interactions with them.

The following paragraph discusses (a) while (b) is detailed in Section 2.2 where we examine Coordination in CMAL, and (c) is covered in Section 2.3.3 which reviews the role of Internal Models.

Credit assignment

With concurrently learning cooperative agents, it is challenging to ascribe the joint reward received from the joint actions to the individual agents based on the role they played in that outcome. There is a whole spectrum of solutions and, at the two extremes of this spectrum, two simple options: *global rewards* and *local rewards*. Global rewards always divide the reward equally among agents. This can lead to the notable failure case of lazy agents since it does penalize free-riders or acknowledge agents that lead the team to success. These drawbacks arise because agents lack feedback that is tailored to their demeanor and this tends to worsen with an increasing number of agents (Wolpert and Tumer, 2001). Global rewards can be impractical to compute in distributed situations where it is challenging to gather information across agents.

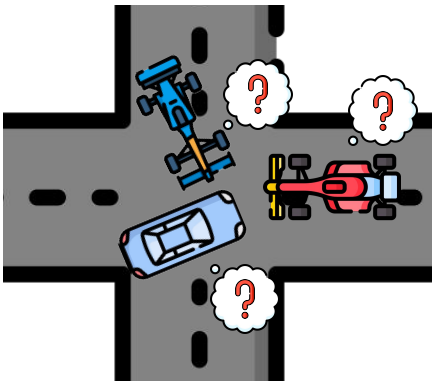


Figure 2.3: The credit assignment problem: each agent wonders if it is the one to blame for the current deadlock situation. Illustration inspired by (Liang and Liaw, 2018).

At the other end of the credit assignment spectrum lie local rewards, where an agent's performance only depends on its own behavior. This prevents laziness but it sometimes encourages greedy behaviors and might prevent cooperation by removing the clear incentive of helping others. In practice, local rewards seem to lead to more homogeneous teams and faster learners while shared rewards are more prone to specialization (Balch et al., 1997; Balch, 1999). This is because the set of "greedy" policies that maximize individual rewards is usually much smaller than the set of policies that can lead to good team performance. Imagine for instance a team learning to play soccer: with local rewards, an agent would only receive a positive reward if it scored a goal. Therefore, it would be straightforward for all the agents to learn to snatch the ball – even from teammates – and selfishly rush for the opponents' net. However, such a team would never evolve sophisticated roles such as goalkeepers and defensive players, or even learn to pass the ball. This effectively reduces the diversity of agents' behaviors and harms team performance.

More sophisticated credit assignment methods try to decompose merit based on agents' behaviors or accountability. In Chang et al. (2003), each agent directly decomposes the global reward by assuming that it combines its own reward with the contributions of the other agents. Agents can extract their individual rewards by modeling the participation of other agents with a random Markov process (Howard, 1960) and filtering it out with a Kalman filter Welch et al. (1995). Some works use *difference rewards* instead and aim a capturing how the team would perform when the agent is removed (Wolpert and Tumer, 2001; Tumer et al., 2002; Tumer and Agogino, 2007; Proper and Tumer, 2012).

More recent MARL methods tend to assign credit based on Q values instead of rewards

as the former are better at capturing an agent's behavior. Inspired by reward decomposition, Sunehag et al. (2017) models the global Q value as a linear combination of agents' individual Q values. Subsequent works have extended this approach to more elaborate decompositions (Rashid et al., 2018; Son et al., 2019). Following the motivations behind difference rewards, Foerster et al. (2018b) proposes a *counterfactual baseline* that marginalizes out the contribution of an agent's action to the global Q value. Another line of work investigates the role of the RL formulation that is used. For example, Tangamchit et al. (2002) advocate for the average reward setting (Mahadevan, 1996) and advise against reward discounting. Indeed, discounting rewards over time can greatly impact credit assignment and prevent cooperation if the effects of an agent's contribution to the reward are delayed.

In addition, credit assignment is also tackled by relying on *social reinforcement* where agents are encouraged to observe and imitate teammates' behaviors or might even receive fractions of teammates' individual rewards (Mataric, 1994a). Those approaches relate to social conventions and will be discussed in the corresponding Related Work Sections 2.3.2 and 2.3.4 that respectively cover Interactions and the use of Shared Incentives.

Finally, credit assignment considerably impacts the learning dynamics of CMAL, and teams with global rewards do not coordinate like teams with local rewards. This is detailed in Section 2.2 on Coordination.

2.1.3 Multi-Agent Communication

Some tasks can only be achieved if agents communicate. In other cases, communication might not be required but benefit learning efficiency and improve performance. This work follows Panait and Luke (2005) popular definition of communication and considers that it refers to agents "altering the state of the environment such that other agents can perceive the modification and decode information from it". Through communication, agents can more efficiently coordinate by sharing information about their state, plans, solutions, the environment, etc.

Most of the time, communication refers to *direct* or *explicit* communication where agents have access to external communication channels or mediums to convey information. Message-passing, signaling, and shared blackboards are some examples of direct communication. Conversely, in *indirect* or *physical* communication, agents do not have a dedicated medium and must repurpose parts of the environment to communicate. This can be done for instance through specific placement of objects or using their bodies.

In practice, communication is always restricted in terms of cost, latency, bandwidth, information loss, locality, etc. Stone and Veloso (2000) explain that it would otherwise simplify MAS to a single agent problem where a central agent could collect everyone's states and decide everyone's actions. However, how this centralized process can emerge through learning is still an open question. When communication is added, the size of an agent's action and observation spaces increase, which can complicate the search for optimal behavior. Conse-

quently, when it has to be learned, communication can hurt performance and even prevent learning (Durfee et al., 1989). For these reasons, most works hard-code communication so as to ease agents' learning process. Some hard-coded communication protocols include current positions (Luke and Spector, 1996; Berenji and Vengerov, 2000a), past experiences or current policies (Tan, 1993), joint utility tables, and joint policies (Berenji and Vengerov, 2000b).

Another line of work is instead interested in learning the communication protocol. Usually, the vocabulary is fixed but unknown and the agents have to associate the words to the correct meanings from trial and error (Yanco and Stein, 1993; Jim and Giles, 2000; Wagner, 2000). Sometimes the vocabulary is instead negotiated and evolved by the agents themselves. Agents collectively define and agree on the words' meanings and the resulting lexicons spontaneously adapt to cover new meanings if needed (Steels, 1996; Steels and Kaplan, 1999).

Similar approaches to *language and naming games* have recently regained popularity in Multi-Agent Reinforcement Learning (Lazaridou et al., 2016; Lazaridou and Baroni, 2020). In those experiments, agents must coevolve and agree on a communication protocol – that is a mapping between words and meanings – in order to solve the task. This can be seen as learned coordination in the communication space. Gupta and Dukkipati (2020) considers another kind of language game in which communication is not used as a referential tool but as a way to persuade. The authors explore a voting game where two candidate agents compete in an election and must convince member agents to gain their votes. Results show that candidates can successfully emerge a language and spread their propaganda to win elections. Interestingly, members of different parties emerge and use different languages such that, eventually, there is an equivalence between an agent's political affiliation and the language it uses.

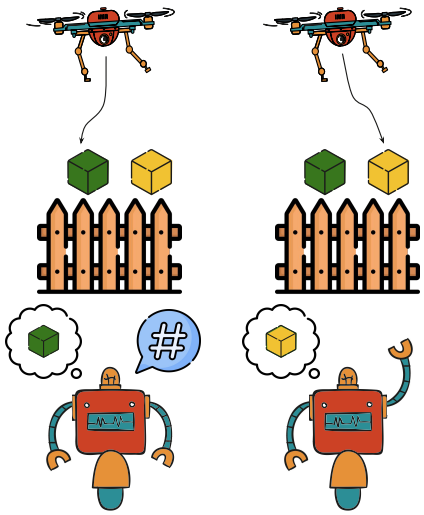


Figure 2.4: A reference game with (a) explicit communication where agents have learned the # symbol refers to the green box, and (b) physical communication where agents have learned that a right arm raised refers to the yellow box.

Other works consider tasks where agents that come up with a common communication protocol perform better. They investigate how agents can evolve communication in order to gain competence toward varied ends such as signaling goals (Lowe et al., 2017) or influencing teammates (Jaques et al., 2019). Most settings consider explicit communication channels between the agents and distinguish between communicative actions (e.g., broadcasting a given message) and physical actions (e.g., moving in a given direction) (Foerster et al., 2016; Lowe et al., 2017; Mordatch and Abbeel, 2018; Jaques et al., 2019).

By using communicative actions that do not have a direct impact on the environment – a notion referred to as *cheap talk* – agents can communicate and develop a communication protocol without having to sacrifice individual performance in the process. This coordination in the communicative action space can enable, to some extent, successful coordination in the physical action space (Farrell, 1987). Yet, *explicit* communication is not a necessary condition for coordination as agents can rely on *physical* communication (Gupta et al., 2017; Mordatch and Abbeel, 2018). Nevertheless, with physical communication, the agent has to trade off how much its action is informative with how efficient it is for the task at hand (Hadfield-Menell et al., 2016), and coordinating becomes much more challenging.

Most of the time, efficient communication rapidly yields rewards, therefore incentivizing agents to communicate and deriving a protocol. Yet, Ackley and Littman (1994) point out that populations can also evolve communication protocols while not being directly rewarded for it as it can present an evolutionary advantage. This is similar to the "kin altruism" or "kin selection" argument of evolutionary theory (Hamilton, 1964).

2.1.4 Applications

We present the main domains of application encountered in the literature. For each of these domains, we list the popular problems being investigated. Figure 2.5 illustrate some of these problems.

Embodied agents

Cooperative Multi-Agent Learning is inherently suited to handle interacting embodied agents and is leveraged in multi-agent robotics, swarm dynamics, computational ecology, population dynamics, wildlife modeling, etc. The most popular applications are Predator-prey pursuit (Benda, 1985; Denzinger and Fuchs, 1999; Zheng et al., 2018), Foraging (Ostergaard et al., 2001), Box pushing (Mataric et al., 1995), simulation and robot Soccer (Kitano et al., 1997; Hester et al., 2010), Cooperative Navigation (Crespi et al., 2002; Robinson and Spector, 2002), Cooperative target observation (Santana et al., 2004; Svennebring and Koenig, 2003) and Herding (Schultz et al., 1996; Potter et al., 2001).

Game-theoretic environments

As presented in Subsections 2.1.5, 2.2.2 and 2.3.1, MAS is often investigated through the lens of Game Theory (GT) and thus it borrows classical problems like Coordination games (Claus and Boutilier, 1998) and Social dilemmas such as the *Iterated Prisoners' Dilemma* and the *Tragedy of the Commons* (Glance and Huberman, 1994; Mundhe and Sen, 2000b).

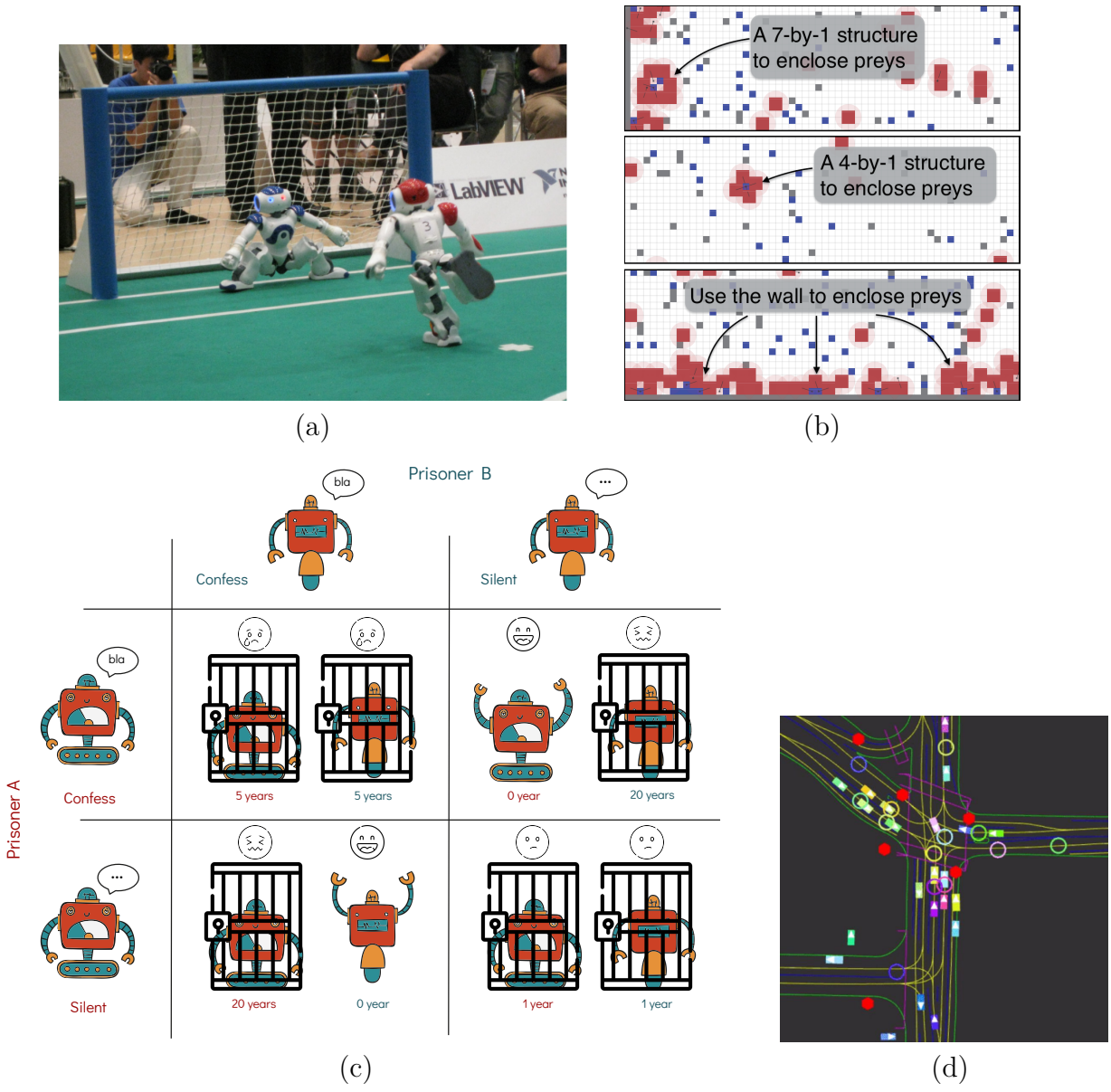


Figure 2.5: Multi-agent learning applications. (a) RoboCup penalty kicks during 2009 semi-finals ©[2010] IEEE (Hester et al., 2010). (b) Predator (red) - Prey (blue) many agents simulations that lead to learned predators' strategies which leverage walls (grey) (Zheng et al., 2018). (c) The Prisoners' Dilemma: prisoners get different jail times depending on whether they remain silent or confess relative to the other prisoner's choice. A prisoner can walk free if it confesses while the other agent remained silent, the silent prisoner gets maximum jail time in that case. (d) A 2D driving simulator that loads and replays arbitrary trajectories and scenes from real-world driving data (Vinitsky et al., 2022).

Real world applications

Many real-world problems in logistics, planning, and constraint-satisfaction require real-time distributed decision-making and are therefore well handled through the MAS paradigm. Examples of real-world applications include Distributed vehicle monitoring (Lesser et al., 1987; Nunes and Oliveira, 2004; Dresner and Stone, 2004) Autonomous driving (Palanisamy, 2020; Chen et al., 2021; Vinitsky et al., 2022), Air traffic control (Steeb et al., 1981), Network management and routing (Weihmayer and Velthuijsen, 1994; Boyan and Littman, 1993), Electricity distribution management (Varga et al., 1994; Schneider et al., 1999), Distributed medial care (Huang et al., 1995), Supply chains (Wooldridge et al., 1996; Brauer and Weiß, 1998), Hierarchical MAS problems such as goods transportation and loading (Fischer et al., 1993; Müller and Pischel, 1994), Models of social interactions (Cederman, 1997; Grand et al., 1997; Grand and Cliff, 1998), and Meeting scheduling (Crawford and Sobel, 1982).

2.1.5 Challenges

We highlight the principal remaining open challenges of CMAL.

Scalability

Arguably, the difficulty in Cooperative Multi-Agent Learning comes from *multi*. Since state-action and behavior search spaces grow exponentially with the number of agents and their network of interaction, learning large, heterogeneous, and interacting MAS remains an arduous challenge. Recent efforts toward this direction notably include the use of parameter sharing for homogeneous teams (Foerster et al., 2016; Gupta et al., 2017; Terry et al., 2020) or attention-based critics to parse large action-state space (Iqbal and Sha, 2019; Jeon et al., 2020). However, work remains to be done in order to "scale up" in terms of *number of agents* and *team heterogeneity*. Additionally, current research assumes very simple agent behaviors and has yet to consider *complex agents* with sophisticated internal states and cognitive processes. From a team point of view, scaling up should both address *changing teams* – how to dynamically add and remove agents – and *changing scenarios* where novel tasks require adapting the team's ability. Last but not least, because the current literature has mostly focused on two-agent problems, it has grown accustomed to being able to predict quite easily the outcomes of interacting agents and behaviors. Yet, the more agents, the more complex the *emergent behavior* and it becomes unfeasible to predict the global effects of a single agent's change in behavior. This makes planning in the behavior space challenging and might even prevent the behavior space from being smooth: a small perturbation in an agent's behavior might drastically alter the emergent behavior of the team (Panait and Luke, 2005).

Definition of team optimality

As previously explained, CMAL deals with multiple agents concurrently learning, adapting to each other, and thus reciprocally changing each other's learning targets: the optimal behavior of an individual depends on the behavior of the group. In return, for a team strategy to be optimal it must be followed by all the agents it relies on (Chalkiadakis and Boutilier, 2003). Panait and Luke (2005) illustrates this by talking about "agents learning in environments where the goalposts are constantly and adaptively being moved". This makes it complicated for individual agents to converge to an optimal policy as their optimal policy is constantly changing with the changing environment. More importantly, it makes it likely for a team to settle on a sub-optimal strategy.

Currently, most of the literature draws ideas from the Game Theory (GT) community and considers convergence to *Nash equilibria* instead of convergence to optimal solutions (Panait et al., 2004b; Wiegand, 2004). A Nash equilibrium is a set of strategies, one for each agent, that has the property that each agent's choice is the best response to the other players' choices. This means that, assuming that other agents will not change their policies, no agent can improve its payoff by modifying its strategy. The most famous illustration of this is the Prisoner's Dilemma (Figure 2.5 (c)). If both agents remain silent, an agent can improve its situation by confessing and walking free. In the situation where one agent confesses while the other remains silent, the silent agent can confess and reduce its 20 years sentence to a five years one. However, if both agents confess, no agent can improve its situation as the agent that would choose to change strategy and remain silent instead of confessing would get a 20 years sentence instead of the current five years. Therefore, according to GT, the Nash equilibrium towards which rational agents converge is the situation where both agents confess. Yet, this is a sub-optimal team strategy. Indeed, if agents would agree on remaining silent and not betray each other, they would get a one year sentence instead of five years imprisonment. While GT does not capture this optimal team strategy, human participants are able to cooperate and achieve it (Axelrod, 1980; Axelrod and Hamilton, 1981).

In light of this limitation, concerns have been raised against the use of Nash equilibria in multi-agent learning, since such "rational" equilibria may be quite far from global team-optimal solutions (Lichbach, 1996; Shoham et al., 2004). Notably, rational agents that explicitly target Nash equilibria fail to coordinate when multiple Nash equilibria exist (Shoham et al., 2004). On the other hand, being optimistic about one's partners and assuming that they will thrive for optimality yields more efficient teams (Claus and Boutilier, 1998; Lauer, 2000; Kapetanakis and Kudenko, 2002b; Panait et al., 2003; Jiang and Lu, 2018). This suggests that CMAL has to move away from purely rational game-theoretic agents and maybe take inspiration from humans – that are shaped by numerous biases such as reputations, laws, guilt, and fear of social punishment – if it aims at team optimality or more accurately modeling human interactions. This will be discussed more in detail in the following Sections 2.2 and 4 where we highlight the importance of shared incentives and

social norms for coordination.

Problem decomposition and learning curriculum

CMAL deals with the large action-state and behaviors spaces, thus, methods often leverage single-agent hierarchical learning in order to decompose an agent's policy into low-level sub-behaviors (or skills) and a high-level skill selection policy. These low-level skills can be hard-coded (Mataric, 1994b, 1998) or automatically learned (Chakravorty et al., 2019). In Roy et al. (2020) we enforce a synchronized and coherent sub-behavior selection mechanism to promote coordination. Yet, instead of just focusing on decomposing an agent's behavior, more work should aim at decomposing multi-agent problems by breaking team strategies into individual agents' roles and behaviors (Stone, 1998). Similarly, single-agent curriculum learning approaches have been adapted to the multi-agent setting. This includes for instance shaping the reward function such that it gradually promotes complex behaviors over simple ones (Balch, 1999) or shifting between tasks based on agents' progress (Zhang and Cho, 1999); however, in multi-agent learning, individual learning mechanisms are strongly coupled. Consequently, the overall progress is more than the mere sum of individuals' progress. Indeed, the team's performance depends on the whole group, and most of the time, agents can only make progress if teammates improve alongside them. Therefore, it is crucial for agents to not only coordinate their policies into an optimal team strategy but also to coordinate their learning. Otherwise, they might never discover and converge to successful strategies. Panait and Luke (2005) depicts this nicely with a team of robots learning to play soccer: in order for an agent to learn how to pass the ball and measure the benefits of such a strategy, its counterpart must have learned to receive the ball, while opponents might ramp-up difficulty by co-adapting and learning to intercept the ball. We further discuss these learning interactions in Subsection 2.3.2 where we focus on the role of interactions in coordinating learning agents.

2.2 Enforcing Coordination in Multi-Agent Systems

As motivated in Chapter 1, coordination is not given in MAS but must be achieved in order to prevent chaotic interactions, meet global constraints, leverage distributed expertise, avoid deadlocks and livelocks, and improve efficiency. This section presents approaches that rely on direct communication to coordinate agents. We start by presenting methods that coordinate procedural agents by structuring their roles and interactions. Procedural agents are agents whose behavior is not learned but predefined and routine-based. Then, after noting that coordination is often thought of as a "negotiation" between agents, we review the relevant literature in game theory-based, planning-based, and human-inspired approaches to negotiation.

2.2.1 Coordination through System Structure

This subsection focuses on the main classical approaches to devise coordination in Distributed Artificial Intelligence. These techniques enforce specific patterns in the structure of the Multi-Agent System. They can be decomposed into two main categories: Organisational Structuring and Contracting (Nwana et al., 1997).

Organizational Structuring

This is the most straightforward approach to coordination as it relies on centralization and *a priori* definition of organizational structures. These structures specify the agents' roles, communication channels, and control flows in order to govern the agents' interactions and resolve the task (Durfee et al., 1987). Classical examples of organizational structures are the *Manager-Worker* and *Client-Server* frameworks which are typically used to allocate tasks and resources to agents. These can be implemented in different ways depending on the communication graph between agents. Usually, one uses a direct communication channel between the manager and each worker. The manager plans and distributes the sub-tasks directly to the workers that send back their results (Figure 2.6).

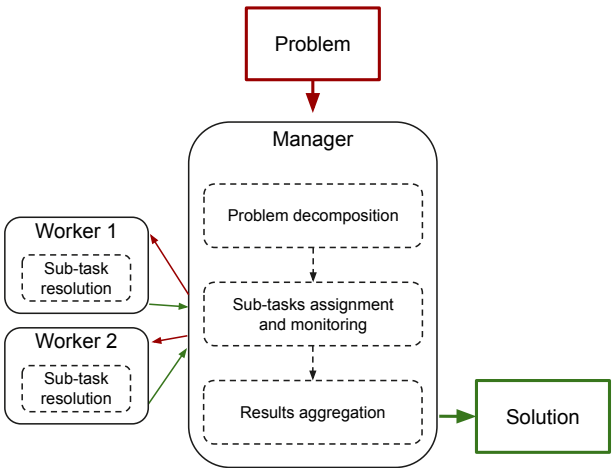


Figure 2.6: The Manager-Worker organizational structure. The manager derives the solution to the problem by decomposing it into subtasks and delegating those to the workers.

Another, more distributed, approach relies on

agents communicating by reading from – and writing to – shared blackboards (Hayes-Roth, 1985). This protocol is depicted in Figure 2.7. The read-and-write operations of the workers are orchestrated by the manager agent, yet this latter can be removed if the task decomposition and sub-task assignments have been done *a priori* (Werkman, 1990; Kearney et al., 1994).

Organizational structuring is well suited for problems where there is an inherent hierarchy, nevertheless, shared blackboard approaches can also be used to coordinate homogeneous or peer agents (Lesser and Corkill, 1983). The limitations of this approach relate to its degree of centralization: most of the coordinating and problem-solving pressure is on the manager agent which creates obvious computation and communication bottlenecks. Also, the shared blackboard implementations require that all the agents use the same semantics to communicate and are therefore often restricted to homogeneous low-level agents. Finally, centralized methods assume there is at least one agent with a global view of the whole system which is often unrealistic in practice (Durfee et al., 1989).

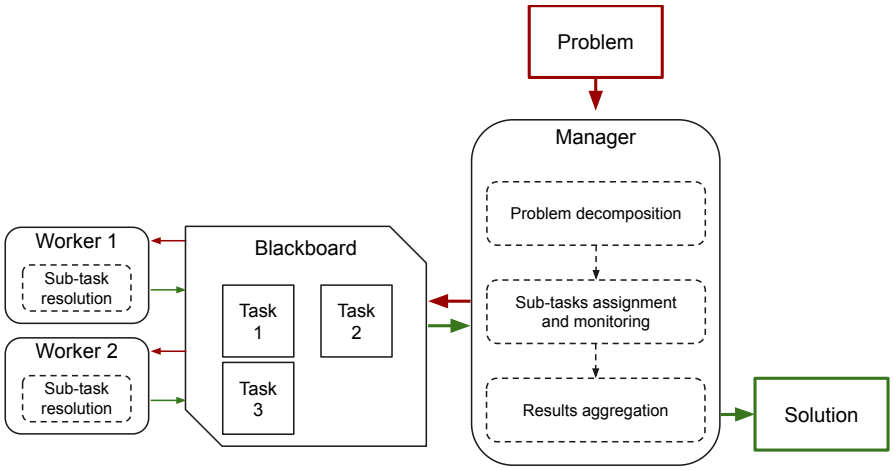


Figure 2.7: Blackboard approach to the Manager-Worker framework. The manager and the worker communicate by reading and writing from a shared blackboard. The contract net protocol has workers compete to get tasks and allows them to turn into a manager to recursively decompose and delegated the subtask they have been assigned.

Contracting

The contract net protocol (Smith, 1980; Davis and Smith, 1983) can be seen as an extension of the shared blackboard scheme above (Figure 2.7). It is a more decentralized and automatic way of decomposing both the problem and the organization structure. This framework assumes a decentralized market structure in which agents can either be:

- a manager that decomposes the problem into sub-tasks and solicits contractors to

achieve them while monitoring the progress of the overall solution, and
- a contractor that carries out a task. Yet, a contractor can recursively choose to become a manager by decomposing the task and proposing the sub-problems to other contractors.

An agent's role is therefore dynamic and the organizational structure can change while the problem is being solved or across problems. The bidding process between managers and contractors follows the ensuing steps:

- a manager proposes a task,
- contractors consider the task in light of their capabilities and obligations,
- contractors place bids on the task,
- the manager gauges the bids and selects a contractor to which the task is dispatched, and
- the manager waits for the results of the task.

This distributed process is used in several settings (Parunak, 1987) and has also been extended to multistage negotiation which enables planners to find acceptable solutions in over-constrained problems (Conry et al., 1988).

The contract net framework proposes a high-level coordination strategy that can efficiently distribute tasks and self-organize groups of agents. Additionally, based on bidding and role self-determination, the dynamical nature of contracting makes it a natural load-balancing and reliable approach: agents can be added and removed, busy agents are not required to bid, etc. (Huhns and Singh, 1994). Still, it remains most suited for applications with a well-defined hierarchical nature that can be decomposed in coarse-grained minimally coupled sub-tasks. Moreover, since the bidding process does not detect nor resolve contradictory demands or conflicting objectives, this framework assumes benevolent and non-antagonistic agents. This assumption rarely holds in real-world scenarios and non-benevolent agents are one of the main motivations for coordination. The contract net approach does not negotiate "what needs to be done" but rather "who is doing it". Some works propose iterative approaches to identify conflicts and reach consensus (Conry et al., 1988), yet they remain highly communication-intensive and are often unfeasible for real-world problems.

2.2.2 Coordination Through Negotiation

As suggested by the last subsection, a significant amount of the literature investigates coordination through the lens of *negotiation*. Indeed, Bussmann and Muller (1992) define negotiation as "the communication process of a group of agents in order to reach a mutually accepted agreement on some matter". More specifically, Sycara (1989) notes that negotiating agents must reason about other agents' beliefs, desires, and intentions, therefore advocating for techniques that:

- represent and maintain belief models,
- reason about other agents' beliefs, and
- influence other agents' intentions and beliefs.

Negotiation has therefore been investigated with methods from a variety of fields such as logic, case-based reasoning, belief revisions, distributed truth maintenance, multi-agent planning, model-based reasoning, optimization, and game theory. It is also a core notion across many research groups working on multi-agent test beds, languages, protocols, and interlingua, as well as purely developmental, cognitive, and sociological research. This subsection focuses on the aspects of negotiation that are closest to coordination and reviews foundational work in this literature. Namely, this section discusses:

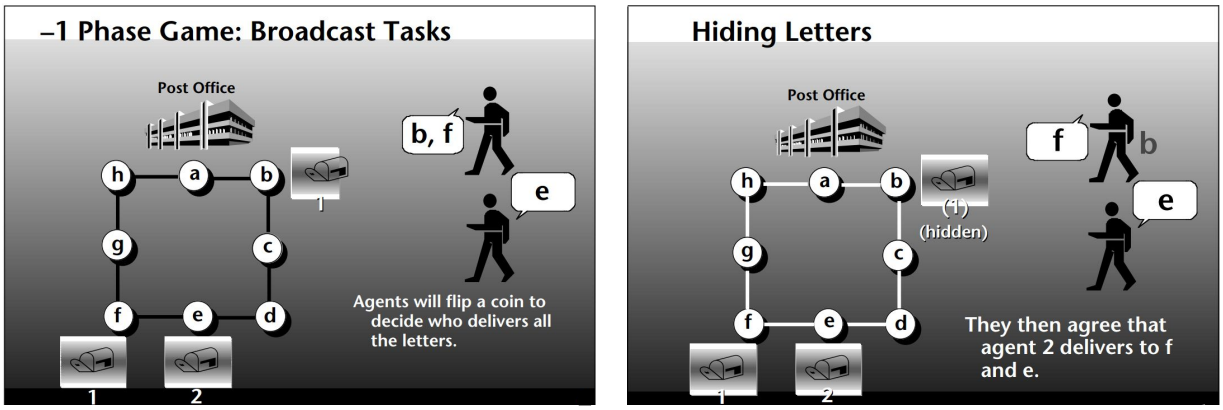
- negotiation in game theory,
- negotiation in multi-agent planning, and
- negotiation in human-inspired AI.

Negotiation in Game Theory (GT)

Rosenschein (1986); Rosenschein and Zlotkin (1994) notably leverage the tools of GT (Luce and Raiffa, 1989) to investigate how rational and autonomous agents can coordinate without relying on pre-defined explicit coordination mechanisms. Instead of considering that agents are benevolent, authors consider that they are negotiating *rational utility maximizers*. The game theoretic concepts of this negotiation approach to coordination are *utility functions*, *deals*, *negotiation strategies*, and *protocols*. Utility functions define an agent's preferences and the goals it aims to archive. Deals are actions that agents can take and are associated with a corresponding utility. Negotiation protocols define the rules through which agents can interact, that is, how to propose, refuse, or accept deals, as well as what happens if agents fail to agree on a deal. Finally, negotiation strategies define how an agent behaves given the set of rules (i.e., the space of deals, utility, and the negotiation protocol).

Rosenschein and Zlotkin (1994) experiment with several domains, protocols, and strategies. In practice, the outcomes of interactions are built into pay-off matrices that are common knowledge to both agents involved in the negotiation. Agents then follow their negotiation strategy to settle on a deal by evaluating the other's offers and proposing counter-offers that maximize their own utility.

Zlotkin and Rosenschein (1990); Rosenschein and Zlotkin (1994) also explore a two-stage setting, in which agents first negotiate a plan and then execute the joint plan. This enables them to investigate how agents can be untruthful and deceptive (withholding information and misinforming other agents) to reach better deals. See Figure 2.8 for a detailed example.



(a) (b)

Figure 2.8: Negotiating deceptive agents, excerpt from Rosenschein and Zlotkin (1994). Agents must deliver letters: agent 1 has to deliver letters at location b and f while agent 2 has to deliver a letter at location e. In order to leave the post office, deliver their letters, and return to the post office both agents will have to cover eight units of distance. (a) agents communicate the locations they must visit and since both agents would have to cover eight units of distance anyway, they agree on a coin flip that decides which agent will deliver all the letters (which also takes eight units). (b) agent 1 lies and only tells about the letter that it must deliver at location f. This would require agent 1 to cover only six units, therefore, there would be no reason for it to walk any additional unit and deliver agent 2's letter at e. Thus, agent 2 agrees to deliver both letters at locations f and e since it does not change the distance it has to cover anyway. By lying and hiding a letter from agent 2, agent 1 has manipulated it into carrying one of its letters. The deceptive agent now only has to deliver the hidden letter to b by walking two distance units.

GT approaches to study coordination suffer from several limitations which complicate their application to real-life settings (Busuioc and Winter, 1995). First, they assume strictly rational utility maximizer agents that follow predefined strategies. Additionally, the pay-off matrices are common knowledge meaning that agents are fully aware of other agents' preferences. These assumptions are unrealistic for truly non-benevolent, partially cooperating agents in real-world settings. Moreover, agents only weigh in current states in their decision, past interactions and future implications are not accounted for. Also, agents are assumed to have the same internal models and capabilities. Finally, building the pay-off matrix for negotiations with many agents and outcomes remains intractable.

Negotiation in Multi-Agent Planning

Adler et al. (1989) argue that planning and negotiation are tightly intertwined as agents need information from other agents to plan effectively. To avoid inconsistent and conflicting behaviors, multi-agent planning approaches propose to collectively build detailed multi-agent plans by specifying all the actions and interactions required to meet the objectives. That way, conflicts can be identified and resolved. Multi-agent planning is either:

- *centralized*: there is at least one agent that possesses a global view of the system, or
- *decentralized*: no one agent possesses a global view.

In centralized planning, a coordinating agent receives all the partial and local plans from the agents in order to analyze them to detect conflicts and inconsistencies. The coordinator then modifies the local plans so as to resolve those incompatibilities and, once the multi-agent global plan is correct, it adds communication commands to the local plans to synchronize the agents' interactions (Georgeff, 1988a,b). There is no need to have a strong hierarchy between agents as one agent can choose to be the coordinator and modify its own local plan in order to resolve conflicts (Cammarata et al., 1988). Conflicts can also be prevented by agents proposing alternative behaviors – that are expected to mitigate the plans' incompatibilities – to the other agents (Jin and Koyama, 1990). Most of the time, the planning process is two-stage: an initial individual planning phase and then the plan coordination (Kreifelts, 1991).

In decentralized multi-agent planning, the novel idea is to equip agents with world models and models of other agents' plans. Through communication, agents can update their local plans – and models of others' plans – until they converge to some global complete and coherent plan (Corkill, 1979; Lesser and Corkill, 1988; Durfee and Lesser, 1988). Due to its iterative approach, decentralized multi-agent planning requires agents to continuously plan and replan, exchanging and processing large amounts of information. This yields complex and resource-intensive protocols even for simple tasks (Corkill, 1979). Yet, this increased complexity is the price to pay to remove the limitation of a centralized system with an agent possessing a global view of the entire system (Huhns and Singh, 1994).

Human-Inspired approaches to Negotiation in AI

Building from the observation that some degree of negotiation is required in almost every human interaction, many negotiation researchers draw inspiration from human negotiation strategies. In practice, to implement these negotiation priors, authors leverage AI techniques such as logic, Case-Based Reasoning (CBR), constraint-directed search, etc.

Sycara (1989) for instance builds on the belief that negotiation is an iterative activity in which human negotiators leverage their experience of past negotiations to inform present and future ones. This leads the author to propose a multi-agent, multiple-issue repeated

negotiation model based on CBR and multi-attribute utility theory. Through CBR, agents can leverage relevant past negotiation experiences and in the absence of those, the model resorts to preference analysis using multi-attribute theory. Negotiated issues are represented by utility curves that are combined to come up with a proposal that maximizes the utility. In addition, Sycara (1989) defines persuasion as the process through which agents can modify others' beliefs, behavior, or intentions, and highlights its importance in negotiations that resolve adversarial conflicts into cooperative interactions.

Werkman (1990, 1992) also believes in the role of past experience and further aims at capturing the importance of having a common background of domain knowledge when negotiating. The author relies on an incremental knowledge-based model where agents can share their perspectives. Typically, knowledge, as well as requested, rejected, or accepted proposals are shared through the use of blackboards. Agents can then leverage this detailed information to improve their proposals in the future. Here the negotiations are three-phase cycles. An agent sends a proposal to the other agent, and the other agent evaluates the proposal. Then this latter either accepts it or sends a counteroffer. If the two agents get into a deadlock, an arbitrator agent generates an alternative offer from the negotiation dialogue and the information network of both agents. While the use of arbitration is novel, it can become a bottleneck – and the same can be said of the centralized blackboard that requires an explicit scheduler to orchestrate the read and write operations.

Sathi and Fox (1989) view negotiation as a constrain-directed search in problem space with negotiation operators. Constraints are dictated by the agents' preferences and the negotiation operators are inspired by human negotiation studies (Pruitt, 1981) and can be used to relax, reconfigure and compose existing constraints into new ones. Negotiation is organized into two phases. First, during the communication phase, information about preferences is exchanged and the initial constraints that define the problem are built. Then, in the bargaining phase, agents negotiate by relaxing the constraints until they reach an agreement and resolve the conflicts. While this iterative approach has been successfully applied to resource allocation, agents are often caught in livelocks of exchanging offers. This is due to the absence of criteria that guide the selection of the relaxation operators.

A similar approach has been proposed by Conry et al. (1988). Authors investigate solutions to distributed constraint satisfaction problems where several agents with limited resources must coordinate toward a common goal. In that situation, the interdependence of the local constraints builds up into a complex set of global constraints.

Finally, more complex, cyclic techniques have been proposed, with notably Bussmann and Muller (1992)'s negotiating framework based on the socio-psychological theory on the eight phases of the negotiation process (Gulliver, 1979). The core idea is that when agents reject a proposal they also list which of their preferences are violated by the proposal. That way, negotiators can update their knowledge about other agents' preferences and come up with better offers.

2.3 Multi-Agent Learning and Coordination

The previous section reviewed the classical approaches to coordination in which agents' decision rules are specified and procedural. This section now focuses on learned behaviors and the underlying mechanisms that enable agents to co-evolve. We care about agents that learn to coordinate and perform well together. Yet, since an agent's progress is coupled with the ones of its teammates, it is essential that they also coordinate their learning. Therefore, this section reviews both *learning coordination* and *coordinating learning*. Specifically, the first subsection discusses the intricacies of concurrent learning and the challenging dynamics that arise. The subsequent subsections respectively delve into the importance of interactions, internal models, and shared incentives in successfully coordinated learners.

2.3.1 The dynamics of concurrent learners

Learning dynamics are complex. In the simplest setting, a single agent experiences a stationary environment and learns from trial and error until it hopefully discovers and settles on a globally optimal policy. When the environment is dynamic and changes over time, the agent can at best track the constantly shifting optimal behavior and try to keep up with the environment's variations. In Multi-Agent Systems the situation is even more intricate as the environment *adapts itself to the agents' behaviors*.

Unfortunately, there are not many tools available to analyze the learning dynamics of concurrently adapting agents. Vidal and Durfee (1997, 1998) propose to monitor specific rates of change during the learning process. They focus for instance on per-agent behavior change rate, learning and retention rates, or the rate at which other agents are progressing. One can then approximate the error in an agent's decision policy throughout the learning process.

Most works that analyze concurrent learning adopt a game-theoretic perspective and draw in particular from evolutionary GT. Notable works have investigated the properties of cooperative co-evolution (Ficici and Pollack, 2000; Wiegand, 2004), including the basins of attractions of Nash equilibria (Panait et al., 2004b), or studied the evolution of policy trajectories from concurrent Q-learning processes (Tuyls et al., 2003; Jan't Hoen and Tuyls, 2004). A crucial concept common to these approaches is the notion of a *Nash equilibrium*, that is a joint strategy – i.e., the global strategy that results from the behavior of each agent – such that no single agent has the rational motivation to unilaterally deviate from it. This means that no agent can get more rewards by changing its behavior while the other agents do not change theirs. Since learners do not usually have control over other agents' behaviors, it is challenging for them to escape Nash equilibria as it would require forming alliances and jointly agreeing on a change of strategy. Consequently, most concurrent learning methods converge to Nash equilibria regardless of the fact that they often correspond to suboptimal team strategies (Panait and Luke, 2005). Tuyls et al. (2003) propose to analyze the Q-

learning dynamics by using the Replicator Equations (Schuster and Sigmund, 1983), an approach commonly used in evolutionary game theory to describe how systems consisting of different strategies evolve over time (see Figure 2.9).

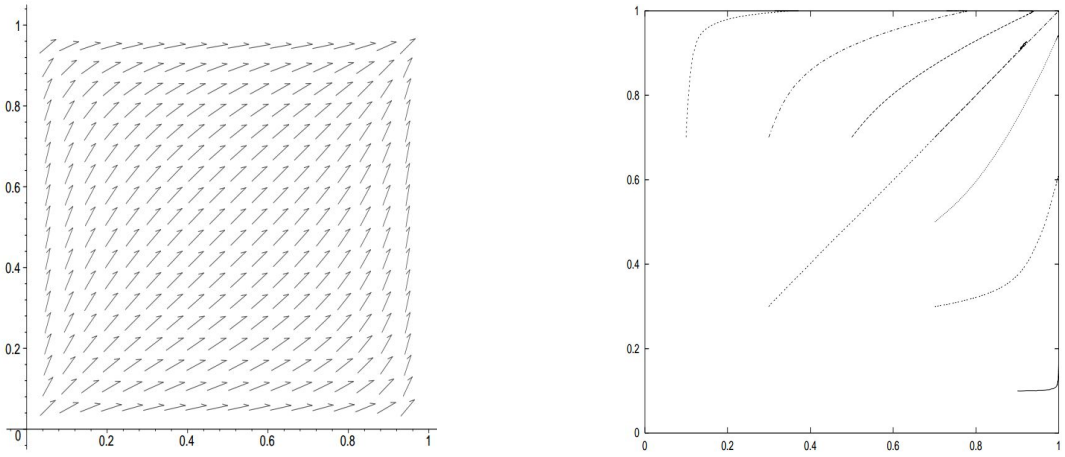


Figure 2.9: Q-learning dynamics in the Prisoners' Dilemma, excerpt from Tuyls et al. (2003). Horizontal and vertical axes are respectively agent 1's and agent 2's probability of confessing. (a) The direction field from the Replicator Equations applied to the Q-learning updates on the agents' strategy: agents converge to the sub-optimal Nash equilibrium of confessing. (b) Converging Q-learning trajectories of the agents' strategies from different initial conditions.

The convergence and learning dynamics of concurrent learners greatly depend on the task's level of cooperation and on the credit assignment scheme of the learning method. In the following paragraph, different settings are discussed:

- the fully cooperative setting, where agents share rewards,
- the competitive setting, where agents' rewards are inversely correlated, and
- general sum games, where agents' rewards might be uncoupled.

Fully cooperative scenarios

In a fully cooperative setting, agents' rewards are positively correlated meaning that increasing one's reward increases the reward of the whole team. There is extensive literature from GT on this setting and it can often be proven that rational agents converge to globally optimal Nash equilibria (Panait and Luke, 2005). Yet, several works point out that, quite disturbingly, this might not be the case in practice with concurrent learners. The corresponding research considers either Repeated Games or Markov Games.

Repeated Games (RGs) model two or more agents that interact by each picking an action. Agents receive some rewards based solely on that interaction: there are no states nor influence from past interactions. Claus and Boutilier (1998) propose two simple repeated games and show that, in practice, RL agents might fail to discover and coordinate to the global optimum. This alarming result holds even when agents know and explicitly reason about the other agent's actions by building Q-value tables for the joint actions. Coordination failure in this ideal situation is worrying since agents usually do not have complete knowledge of the other agent's behavior. The experiment is described more in detail in Figure 2.10.

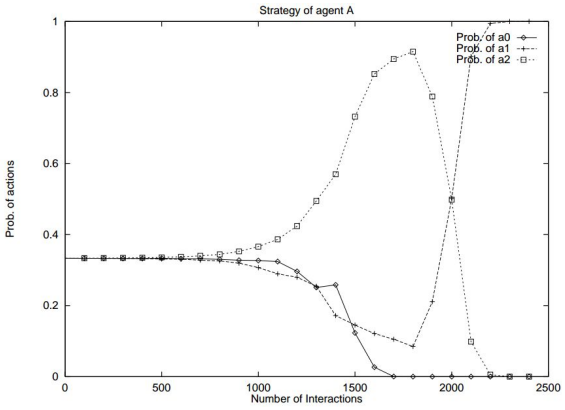
Lauer (2000) present an optimistic approach that updates an agent's Q-table by estimating the best cooperation possible to the corresponding action. The author proves the method's convergence to the optimal strategy in deterministic repeated games. After pointing out the flaws of this approach for stochastic games, Kapetanakis and Kudenko (2002a,b) propose an adapted exploration strategy and report improved cooperation in this new setting.

Brafman and Tennenholtz (2002) derive a stochastic sampling procedure that guarantees the convergence to optimal Nash equilibria. Nevertheless, this approach relies on restrictive assumptions conflicting with the concurrent learning paradigm. Indeed, the method structures the agents' interactions and learning *a priori*: it defines and enforces a joint exploration phase followed by a joint learning phase at the end of which each agent settles for the behavior that yielded the best outcome. If anything, this illustrates the need to *coordinate exploration and learning* in CMAL.

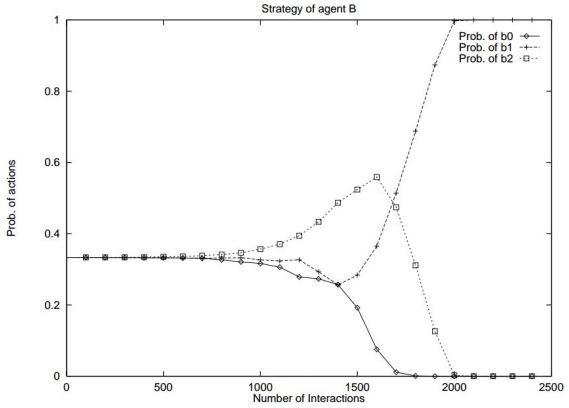
Markov Games (MGs) extend RGs by incorporating the notion of states: at any point in time, the game is in a given state. The next state is sampled stochastically from a transition function that depends on the current state and the agents' interaction. The rewards are now based on both the current state and the current interaction. If there is a single state, a MG reduces to a RG, while with only one agent a MG is a Markov Decision Process. Most works done in MGs consider general-sum games and are therefore presented in the next subsection. A notable exception, work by Wang and Sandholm (2002), presents Optimal Adaptive Learning, an algorithm guaranteed to converge to global Nash equilibria in fully cooperative stochastic games with a finite number of actions and states. To avoid converging to suboptimal equilibria, the method emulates "virtual games" at each state such that solving them eliminates potential suboptimal Nash equilibrium. Unfortunately, this approach can be prohibitive as the number of virtual games to solve scales exponentially with the number of agents.

	a_0	a_1	a_2
b_0	11	-30	0
b_1	-30	7	6
b_2	0	0	5

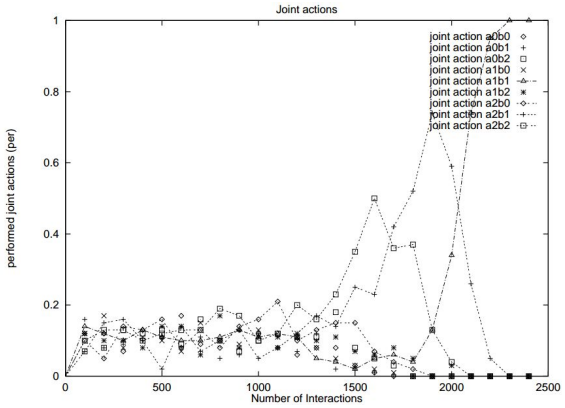
(a)



(b)



(c)



(d)

Figure 2.10: Suboptimal coordination in a fully cooperative setting, excerpt from Claus and Boutilier (1998). (a) the climbing game in which the optimal joint strategy is (a_0, b_0). (b) and (c) respectively agent's A and B strategy throughout learning. (d) joint strategy throughout learning. Agents start by selecting actions uniformly and the severe -30 penalties that can arise for actions with indexes 0 and 1 drive agents toward the non-equilibrium strategy (a_2, b_2). Yet, agents "settle" but continue exploring around this point and agent B finds b_1 more attractive (provided that A continues to choose a_2 most of the time). Once the strategy shifts to (a_2, b_1), agent A tracks B's move and realizes that a_1 is a better response. Therefore, agents eventually converge to (a_1, b_1) but never reach the optimal strategy (a_0, b_0) because it would require agents to *simultaneously* switch strategies and avoid the discouraging -30 penalties.

Decentralized Partially Observable Markov Decision Processes (Dec-POMDPs) extend Markov Games by considering that agents do not have access to the global state of the game but only have incomplete and partial *observations* of it. If there is only one agent, a Dec-POMDP becomes a Partially Observable Markov Decision Process (POMDP). A Decentralized Markov

Decision Process (Dec-MDP) is a special case of Dec-POMDP in which, at any point in time, the global (hidden) state can be uniquely determined from the current set of observations of the agents. Note that a Dec-MDP is more challenging than a MG as in the latter all the agents directly observe the global state of the game. In terms of complexity, MDPs and POMDPs are respectively P and PSPACE-complete (Papadimitriou and Tsitsiklis, 1987) while Dec-MDPs and Dec-POMDPs are both NEXP-complete (Bernstein et al., 2002). For more discussions on computing complexity, the reader is referred to (McMahan and Gordon, 2007; Deng et al., 2023).

In fully cooperative Dec-POMDPs, Peshkin et al. (2001) showed that their distributed RL method converged to local optima but that were not necessarily Nash equilibria. Nair et al. (2003) observed that exponential training speedups could be achieved by training agents one at a time, keeping the policies of the other agents fixed.

Cooperative-Coevolutionary Algorithms (CEA) were originally proposed to decompose a problem and concurrently search for solutions to the sub-problems (Potter and De Jong, 1994; Potter, 1997). They are well suited to tackle cooperative RGs with many works that have tuned the algorithms to improve performance Bull (1997, 1998); Wiegand et al. (2001). Nevertheless, most works have focused on analyzing agents' convergence to Nash equilibria rather than investigating how to derive teams that find globally optimal solutions (Wiegand et al., 2002b,a; Wiegand, 2004). Interestingly, Panait et al. (2003, 2004a,b) note that, when applied to the cooperative setting, standard CEA methods tend to sacrifice performance in favor of *balance*. In other words, agents are driven to co-adapt to teammates regardless of the latter's poor performance. This may be counterproductive and agents should rather assume that teammates will rationally strive to become ideal collaborators in order to aim for the corresponding optimal behavior. Indeed, the performance of CEA algorithms is greatly improved if agents are evaluated with teammates that approximate their best possible collaborators. Yet, estimating these "good partners" is non-trivial, and methods usually rely on good teammates from past iterations. For instance, Gordin et al. (1997); Puppala et al. (1998) incrementally build a "hall of fame" repository by keeping the best teams evolved so far. Authors then evaluate individual learners by selecting teammates from the repository. Blumenthal and Parker (2004) do something similar by having learners periodically provide demonstrations representative of their current policies to other learners so those latter can use it to evaluate their own training.

General sum games

In General sum games, there is no specific structure for the agents' rewards or credit assignment. Consequently, one agent can increase its reward without the other agents increasing theirs and the increase in one agent's reward can sometimes even yield a decrease in another's. This setting can thus result in highly non-cooperative situations.

General sum games are usually represented in two forms. Either extensive-form – as a tree structure – which best captures agents acting in sequence from imperfect information. And normal-form – as a pay-off matrix – which is most relevant for simultaneous moves from perfect information (Nisan et al., 2007). Few works study the dynamics of concurrent learners in extensive-form games – with the exception of Tesauro and Kephart (2002) – and most of the literature – starting from Littman (1994) – focuses on normal-form games.

Bowling and Veloso (2000) start by analyzing a number of GT and RL approaches to Markov Games and discuss the different assumptions that are made. Building from this, several works investigate the importance of modeling other agents in order to reach better equilibria. For example, Hu et al. (1998); Bowling (2000); Hu and Wellman (2003) extend distributed RL methods so that agents also learn Q-values tables for the other agents. From these, agents can estimate the actions that others might select. Nagayuki et al. (2000) choose to directly approximate the policies of other agents instead of relying on Q-values.

Suematsu and Hayashi (2002) highlight a crucial limitation of previous works by noting that agents merely try to reach Nash equilibria and assume that the other agent will also move towards it. Instead, agents should explicitly adapt to others and influence their behaviors toward optimal strategies. The authors observe that current methods would fail in situations where the other agents are following fixed policies. Subsequent works thus focus on proposing different solutions based on how the other agents could adapt. Suematsu and Hayashi (2002) for instance proposes a method that reaches Nash equilibria if other agents are adaptable and strive to do so as well. Otherwise, in the case of fixed teammates, the agent aims for an optimal response policy. Similarly, Littman et al. (2001) proposes an algorithm where agents look either for coordination or adversarial equilibria depending on whether their teammates are cooperative or competitive.

Finally, Bowling and Veloso (2001) focus on the notion of optimality and propose two desiderata for concurrent learners: rationality and convergence. Rationality states that the agent should converge to its corresponding optimal policy when other agents have converged to stationary behaviors. Then, convergence dictates that all agents should eventually converge to stationary strategies. Greenwald et al. (2003) propose to enforce rationality by drawing from *Nash correlated equilibria* (Nisan et al., 2007) – an alternative solution concept to Nash equilibria – and proposes Correlated-Q. The authors demonstrate empirical convergence of the method. At the cost of increased centralization, this approach allows agents to discover and use strategies that require them to pick coherent options to break the symmetries in the problem. Indeed, in situations where there exist multiple Nash equilibria, especially if these are unfair and favor one agent over the other, it might be challenging for agents to coordinate and agree to select the same equilibrium. Correlated equilibria enable agents to coherently alternate between different equilibria and fairly share payoffs.

To better understand this we need to discuss additional equilibrium concepts. So far, we have discussed *pure* Nash equilibria in which agents use pure strategies which means that they pick actions deterministically. Agents can also act stochastically and have a probability distribution over their actions: these are called mixed strategies. A *mixed* Nash equilibrium is a situation where no agent has the incentive to change its mixed strategy (i.e., its probability distribution over actions) if no other agent does it. Finally, a *correlated* Nash equilibrium is a situation in which agents have agreed *a priori* on a set of joint strategies and their likelihood. Before acting, a joint strategy is drawn and each agent receives an observation indicating the corresponding action it should take (but not the whole joint strategy). Knowing what the other agent is likely to do (because it knows the potential joint strategies corresponding to the observation), the agent should have no reason to deviate from the move suggested by the observation. The *Bach or Stravinsky Game* (Table 2.1) illustrates this well. There exist two (unfair) pure Nash equilibria: (Bach, Bach) and (Stravinsky, Stravinsky) that respectively favor Player 1 and Player 2. There is also a mixed Nash equilibrium: Player 1 goes to listen to Stravinsky one-third of the time while Player 2 does it two-thirds of the time. Indeed, knowing that this is the strategy of the other agent, no player has the incentive to deviate from it. Let us look at it from Player 2's perspective: it knows Player 1's strategy and wants to figure out with which probability p it should go to listen to Stravinsky. Its expected payoff is

Player 1 \ Player 2	Bach	Stravinsky
Bach	2, 1	0, 0
Stravinsky	0, 0	1, 2

Table 2.1: The *Bach or Stravinsky Game*'s payoff matrix. Two persons want to meet at a music recital but one of them (Player 1) prefers Bach while the other (Player 2) prefers Stravinsky. Which event should they choose?

$$\begin{aligned} &p \times 1/3 \times u_2(\text{Stravinsky, Stravinsky}) + (1-p) \times 2/3 \times u_2(\text{Bach, Bach}) \\ &= p \times 1/3 \times 2 + (1-p) \times 2/3 \times 1 = 2/3. \end{aligned}$$

This is independent of p so it has no incentive to change strategy (u_i is agent i's utility as given in the payoff matrix). Similarly, we can verify that if Player 2 goes to Bach a third of the time, Player 1's payoff is constant regardless of its strategy. Therefore, Player 1 and Player 2 going to Bach respectively two-thirds and one-third of the time is a stable and fair solution (both players have an expected payoff of 2/3). Yet, this mixed equilibrium is not very interesting since both agents get in expectation less payoff than they would in an unfair pure equilibrium. A more interesting solution is the following correlated equilibria: agents agree on two joint strategies, either they both go to Bach or they both go to Stravinsky. A strategy is picked uniformly at random and each player gets an observation hinting at what it should do. There are only two joint strategies and it is trivial to show that agents do not deviate from it: for instance, if Player 1 is suggested a place (Stravinsky for instance) it

knows that Player 2 will go there (both going to that place is the only joint strategy that can indicate that place to Player 1) and it should go there to get the positive payoff. That correlated equilibrium is fair and both agents get $1/2 \times 2 + 1/2 \times 1 = 3/2$ expected payoff. Note that a correlated equilibrium requires pre-defining joint strategies, sampling those, and having agents observe their expected roles.

Nowe et al. (2001); Peeters et al. (2004) also has agents alternate between to unfair equilibria to reach a more equitable outcome repartition. The authors propose a method in which agents communicate their collected payoff. If the situation is unfair, meaning that there is a discrepancy between the agents' payoffs, the currently played actions are removed from the agent's action space, and agents must find a new equilibrium.

Competitive learning

While the focus of this literature review is on cooperative settings, learning pathologies present in competitive environments stem from similar dynamics and provide interesting insights. Additionally, as discussed more in detail in Subsection 2.3.2, competitive learning is often used as a way of "training" agents by pitting them against one another which forces them to improve in order to overcome their opponents. Think for instance of two agents learning the game of Checkers from scratch by playing against each other: they learn at the same pace and complexity gradually ramps up as they improve.

There are three main learning pathologies that can arise from competitive learning: *loss of gradient*, *the Red-Queen effect*, and *cyclic behaviors*.

Loss of gradient happens when one learner starts to dominate the other one. If regardless of what it does, the dominant player always wins, while the inferior player always loses, no matter what it tries, none of the agents is receiving any feedback that can be used to improve (Pollack et al., 1997; Watson and Pollack, 2001). This is closely related to the "laziness" issues in cooperative learning. Wiegand and Sarma (2004) examine this behavior in a coevolution domain in which some agents can learn faster than others because they are provided with better learning opportunities. They show that using spatially embedded Cooperative CEA alleviates the loss of gradient pathology and improves learning performance. Here, spatial embeddings are a way of artificially coupling the learners' progress by constraining the selection of new generations. Learners' populations are distributed across some predefined virtual space and each learner evaluates its individuals based on how they perform with the other learners' individuals in the neighboring space.

The Red Queen effect refers to the difficulty of monitoring a learner's progress because its fitness depends on other learners' abilities (Cliff and Miller, 1995). Back to the two Checkers learners: initially, they draw on most of the games but then, after some training, the first agent consistently wins against the other one. From this, little can be said about the

progress of the players. It could be that both improved but the first one improved more, or that the first player's performance remained constant but the second player's skills dropped. However, it could also be that *both players got worst* at checkers but the second one more.

The Red Queen effect also occurs in cooperative settings. For example in symbiotic relationships, the mutual benefits are only maintained if both agents continuously adapt and evolve. This can lead a successful pollinator species to go extinct because the plant species it was relying upon could not adapt and disappeared (Herre et al., 1999; Gao et al., 2015). A related pathology occurs with cooperative multi-agent learning and local rewards: even if every agent improves with respect to its individual reward, there is no guarantee that the team is improving as a whole. Imagine a task where all agents must reach a target location and each agent is rewarded based on its proximity to the target. All agents might learn to get close to the target, yet accidentally block each other and prevent some agents from actually reaching the target.

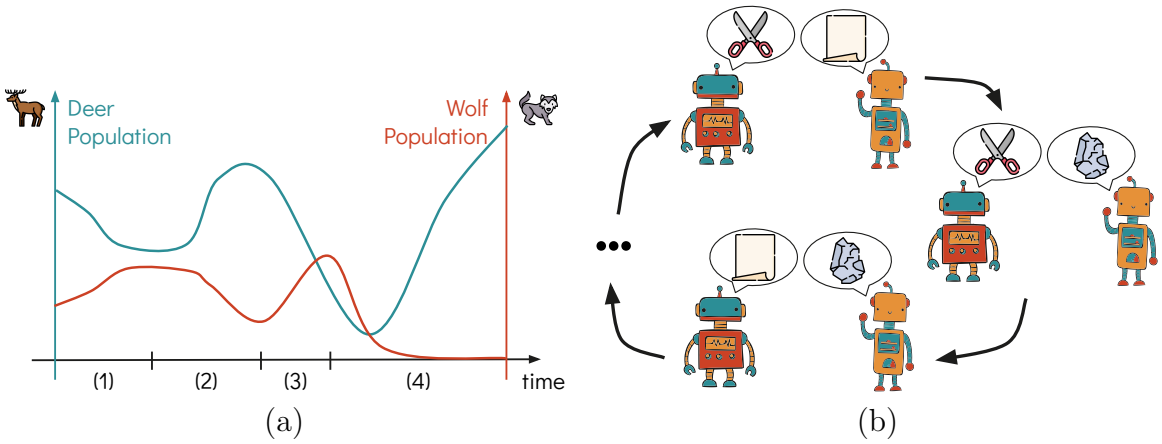


Figure 2.11: Illustrations of common multi-agent learning pathologies. (a) the Red-Queen effect: (1) wolves hunt deer, (2) deer evolve to be more agile and escape wolves, (3) wolves evolve to hunt more efficiently in packs, and (4) humans start to hunt wolves while deer become less agile. Looking at population curves alone does not give the full picture as one cannot know if the variations are due to one specie's improvement or the other's collapse. For instance, the deer specie was about to disappear because their predator became too efficient at hunting them (3). Yet the intervention of a third species removed wolves and deer now strive, however, deer skills did not improve. (b) Cyclic behaviors: agents keep alternating between dominant and dominated strategies indefinitely.

Cyclic behaviors can be seen as learning livelocks – while the loss of gradient pathology was more of a learning deadlock. Cyclic behaviors are likely to occur if there exist non-transitive relationships in the problem setting or in the solution strategies. The Rock-Paper-Scissor game is a good example of a non-transitive cycle: paper beats rock, rock beats scissor and scissor beats paper. This means that two competing agents can easily be trapped in a cycle

where they keep co-adapting indefinitely without any overall progress (Cliff and Miller, 1995; Rosin and Belew, 1997; Ficici and Pollack, 1998).

Cyclic behaviors also occur in cooperative learning, an example of that is the cyclic emergence of "leaders" that try to solve the task on their own and then "pass the lead" to another agent. A well-coordinated team where all the agents participate at any point in time is often more efficient. Picture a task where two agents must find an object and receive a shared reward. Agents might end up taking turns in who is the "leader" that searches for the object while it would be more efficient for them to simultaneously search for it. This case is related to the "laziness" pathology that is exacerbated with shared reward. However, "cyclic leaders" also occurs with local reward where agents alternate in taking the lead because they seek extra rewards, often at the expense of the team's performance. Imagine a four-legged ant robot where each learner controls a different limb and is rewarded by the forward progress of this limb – the team objective being to move the robot forward. Agents might alternate leads as "dominant" agent that makes the robot turn so the "dominant" leg is forward and covers more ground. Turning the robot that way to have a leg forward might be detrimental to having the whole robot move forward as much as possible.

The risk for loss of gradient learning failures is even more severe – and conflated with the Red-Queen effect – in domains that are a mixture of cooperation and competition. Such settings include tasks where agents must learn to cooperate as a team in order to compete against an opposing squad. Luke et al. (1998b) for instance trains a team to play soccer against opponent teams and highlights the need for coordinating the agents' learnings. Indeed, in order for an agent to learn how to pass the ball, its teammates must have learned to receive the ball while opponents might simultaneously co-adapt and learn to intercept passes. While few works have investigated how to decompose these "learning dependency graphs", Guestrin et al. (2002) note that the action domains of agents are often coupled. Therefore, they propose to decompose the Q-values according to a heuristical *coordination graph* that describes which agents have to interact together in order to reach a solution. This provides a middle ground between having to learn the whole joint value tables and relying on independent individual value tables. Along those lines, Makar et al. (2001); Ghavamzadeh and Mahadevan (2004) propose a more hierarchical approach that focuses on coordinating the agents' high-level behaviors rather than their primitive actions.

This subsection has discussed the complex dynamics of concurrent learning, highlighting common pathologies and problems. In particular, it remains a challenge for agents to discover optimal team strategies and to converge to the corresponding equilibria. We draw from the literature that points toward the crucial need for *agents that coordinate their learnings*, and review in the following subsections three key elements for devising coordinated learners.

2.3.2 Interactions in Multi-Agent Learning

Interactions are central to multi-agent learning and they result in challenging non-stationary learning environments from the agent's point of view (Lowe et al., 2017). This makes the problem of coordinating the agents into optimal collective behaviors a NEXP-hard one (Bernstein et al., 2002). Instead of focusing on the challenges that arise from multi-agent interaction (which has been covered in the previous Subsection 2.3.1), this section investigates how multi-agent interactions can be leveraged to improve coordination and learning. First, we discuss how multi-agent competition can design an automatic learning curriculum. Then, we review how learners can leverage their interactions with other agents to improve. After that, we present how agents can use interactions to influence their teammates. Finally, we conclude with the challenges of evaluating interacting agents.

Automatic Curriculum Learning

There is an important body of work that investigates how interactions between learning agents are a way of progressively tuning up the task difficulty so as to guide training. This can be seen as a multi-agent approach to curriculum learning. *Curriculum learning* proposes to decompose and sequence tasks and data samples into a curriculum. That way it is possible to train agents on problems that would be too challenging to learn all at once (Elman, 1993; Sanger, 1994; Rohde and Plaut, 1999; Bengio et al., 2009; Narvekar et al., 2020; Portelas et al., 2021; Forestier et al., 2022). This approach builds on the notion of *optimal learning difficulty* or *Goldlilocks zone* that refers to tasks whose difficulty is *just right* so as to challenge the agent and drive it to improve (Kidd et al., 2012; Wilson et al., 2019). Too easy a task won't require any progress since the agent can already solve it. On the other hand, one might never manage to solve tasks that are too difficult and thus fall short of learning signals to follow. The Goldilocks learning zone is conceptually close to the *zone of proximal development* proposed by (Vygotsky, 1934) to describe the tasks that a learner cannot solve alone but can manage if provided with guidance. Crucially, the optimal learning difficulty is a function of a learner's ability and difficulty must increase with skills. Curriculum learning aims at designing a progression of tasks that track an agent's Goldilocks learning zone. By gradually increasing difficulty, the agent can build upon its existing knowledge and skills to improve and take on more and more challenging problems (see Figure 2.12).

In the multi-agent setting, the same considerations hold true. For instance, with two agents learning to play the game of Checkers it is often desirable for them to learn at the same pace, with difficulty gradually ramping up as they improve. Self-play (Samuel, 1959; Tesauro, 1991), where a single learner plays a multi-agent game against itself, has been widely and successfully used for that purpose over the years. Notable examples include Checkers (Samuel, 1959), Backgammon (Tesauro, 1991), Poker (Kendall and Willdig, 2001), Hanabi (Bard et al., 2020), Chess, Shogi, Go (Silver et al., 2018) and video games (OpenAI et al., 2019).

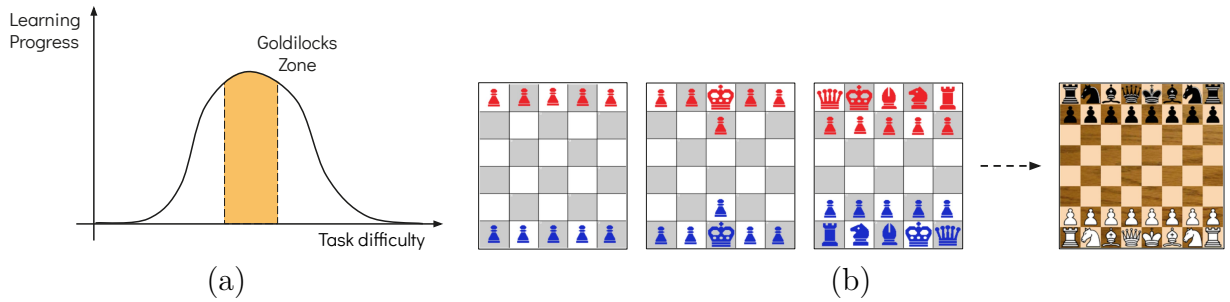


Figure 2.12: Illustration of the curriculum learning approach. (a) The Goldilocks learning zone: if the task is too easy there is nothing to learn while if the task is too difficult we cannot make progress on it. One needs the difficulty to be "just right" and in tune with current skills. (b) Different subgames in the game of Chess are used to form a curriculum for learning the full game of Chess by progressively increasing the task's complexity and tracking the agent's Goldilocks learning zone throughout training. Image (b) is an excerpt from Narvekar et al. (2020).

Automatic learning curricula can also arise from competitive environments involving multiple learners. Bansal et al. (2017) highlight how competition naturally devises a learning curriculum and can foster the emergence of very complex behaviors in otherwise simple environments. Jaderberg et al. (2019); Baker et al. (2020) extend this idea to mixed cooperative-competitive environments. In Baker et al. (2020) for instance, agents play a team version of the "hide-and-seek" game and can interact with objects present in the environment. Authors show that such a simple setting can lead to the emergence of elaborated team strategies and sophisticated tool use (see Figure 2.13). Yet, these works point out that, in order to avoid the loss of gradient pathology discussed previously, it is important to keep the opponents' level of skills balanced and in check. One way to do this is to evaluate the current learned agents against a whole population of opponents, for example, considering past versions of the opponent. Additionally, this tends to select robust behaviors that are resilient against varied opponents.

As mentioned above, self-play is also commonly used to train agents in the cooperative setting with either a single (Bard et al., 2020) or multiple learners (OpenAI et al., 2019). Interesting variations on self-play have been investigated for specific cooperative settings. Other-play (Hu et al., 2020) for instance aims at devising agents for Zero-Shot Coordination (ZSC). ZSC refers to agents that are able – without additional retraining – to perform with partners they have never seen before. Another line of work (Lerer and Peysakhovich, 2017), modifies self-play to train agents to maintain coordination in social dilemmas. Social dilemmas refer to situations in which agents are tempted to egoistically increase their individual outcomes to the detriment of global welfare. It is therefore challenging for agents to maintain cooperation: they must start by cooperating but avoid being exploited while trying to return to mutual cooperation if that occurs.

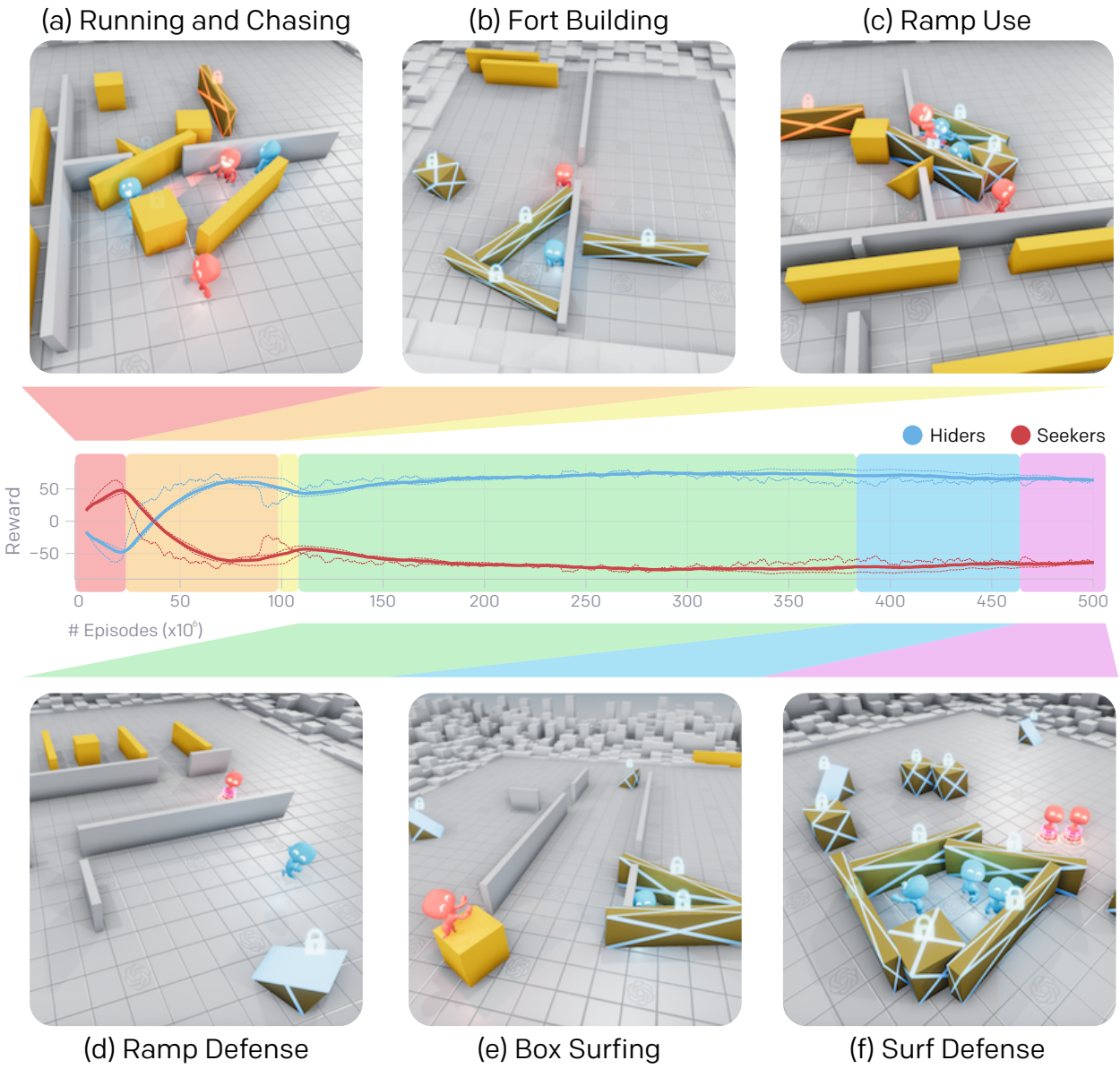


Figure 2.13: Emergent Skill Progression From Multi-Agent Autocurricula. Through the reward signal of hide-and-seek (shown on the y-axis), agents go through 6 distinct stages of emergence. (a) Seekers (red) learn to chase hiders, and hiders learn to crudely run away. (b) Hiders (blue) learn basic tool use, using boxes and sometimes existing walls to construct forts. (c) Seekers learn to use ramps to jump into the hiders' shelter. (d) Hiders quickly learn to move ramps to the edge of the play area, far from where they will build their fort, and lock them in place. (e) Seekers learn that they can jump from locked ramps to unlocked boxes and then surf the box to the hiders' shelter, which is possible because the environment allows agents to move together with the box regardless of whether they are on the ground or not. (f) Hiders learn to lock all the unused boxes before constructing their fort. Image and captions are excerpts from Baker et al. (2020).

Social Learning

The benefits of automatic curriculum from competitive learning have drawn researchers to investigate similar mechanisms in the absence of competition. Here, we regroup these approaches under the umbrella of *social learning* (Jaques, 2019).

Often, the motivation is to derive behaviors that are robust, handle a wide range of tasks, and are resilient to domain changes. In that case, most approaches train a learner to solve tasks that are adversarially adapted by a competing learner. This is sometimes referred to as "host-parasite" co-evolution (Hillis, 1990; Cliff and Miller, 1995; Harvey et al., 1997). It is challenging to generate tasks that are (1) relevant to the problem, (2) whose difficulty is adapted to the current abilities of the agent, and (3) solvable. To tackle this, Dennis et al. (2020) propose Unsupervized Environment Design (UED) where an environment-agent picks the parameters of procedurally generated environments that are then used to train a protagonist-agent. In parallel to the protagonist-agent, an antagonist-agent is trained to solve the same generated domains. The difference in performance between the antagonist and protagonist agents is used to guide the environment design towards settings that are solvable by the antagonist but challenging for the protagonist. In Gur et al. (2021), authors extend this setting to generating compositional tasks and training a whole population of agents instead of a single protagonist-antagonist pair.

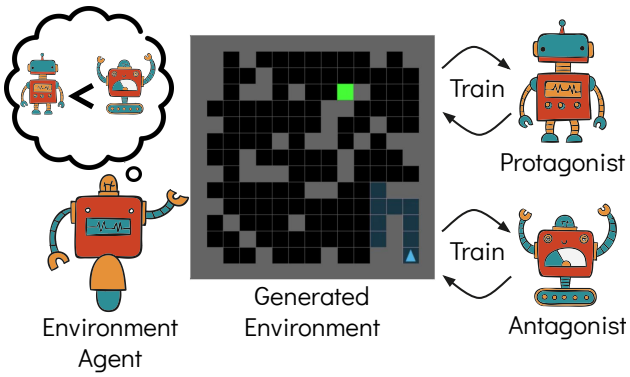


Figure 2.14: Unsupervized Environment Design. An environment agent generates the environments in which a protagonist and an antagonist agents are trained. The goal of the environment agent is to generate tasks in which it believes the antagonist will perform better than the protagonist. That way the protagonist is trained on tasks that are solvable but that it finds challenging.

The idea of leveraging the interactions between agents to improve learning has also been investigated in Imitation Learning (IL) and Inverse Reinforcement Learning (IRL), where a demonstrator is trained to generate trajectories that might not be optimal but that would best teach an apprentice agent (Hadfield-Menell et al., 2016). Similarly, Colas et al. (2020) explores the benefits of having a social partner provide language descriptions of a learning agent's activity. Finally, learners can benefit from interacting with other agents even if the latter are not actively trying to teach them. Ndousse et al. (2021) report that agents trained in the presence of other learners – and/or experts – learn faster, better, and develop more generalizable policies. Mataric (1994a) refers to this phenomenon as *observational reinforcement*: it rewards agents for observing and imitating teammates. This can help reproduce

rare behaviors and improve the overall team performance (more on this in Subsection 2.3.3). Other forms of *social reinforcement* include *vicarious rewards* through which agents receive portions of the rewards received by neighboring agents in order to diffuse individual rewards in a tradeoff between local and global rewards.

Interactions as influence

Agents must leverage interactions, whether through physical or communicative actions, to influence others. Early on, Chalkiadakis and Boutilier (2003) proposed a method in which agents reason about how their actions will influence the behavior of other agents. By influencing each other, agents can eventually coordinate their policies toward equilibria. Similarly, Jaques et al. (2019) draw from the notion of Causal Influence (Collins et al., 2004) and reward individual agents for producing actions or messages that will impact the behavior of their teammates. Authors show that using this intrinsic motivation leads to enhanced cooperation and increases the collective performance of the team. Additionally, they note that it can lead to the emergence of coherent communication protocols when used with explicit communication channels.

Instead of restricting influence to considering the impact of an agent's actions on another's, Foerster et al. (2018a) propose Learning with Opponent Learning Awareness (LOLA) where agents directly consider how their learning influences other agents'. In practice, LOLA's policy update optimizes for one-step lookaheads of opponent learning, which mean that it adapts to – but also actively influences – the other agent's next policy update. Authors show that shaping the learning of other agents allows for the emergence of cooperation out of self-interest, consequently improving on Nash equilibria. For instance, the method leads to Tit-for-Tat strategies in the iterated prisoners' dilemma, this means that agents start by collaborating (and remaining silent) and then mimic what the other agent does, so they defect (i.e., confess) after being betrayed but forgive and agree to collaborate again if the other agent switches back to collaboration. Tit-for-tat strategies are based on the concepts of retaliation and altruism and, while being robust against exploitation, they can lead to much better outcomes in the iterated prisoners' dilemma.

Finally, a very efficient way of influencing other agents is to set their objectives, such as their rewards or goals. Ahilan and Dayan (2019) considers Feudal Learning based on a hierarchical decomposition of the problem where a manager agent can set the goals and rewards of other agents. Yang et al. (2020) extends this idea to a more horizontal and distributed setting by allowing each agent to give rewards directly to other agents.

Evaluating teams of agents

It is challenging to evaluate the performance of interacting agents when they are trained together because the performance of an agent – and of the team – depends on the performance

of its teammates (cf. the Red-Queen effect in Subsection 2.3.1). Nevertheless, evaluating an agent with the teammates it has been trained with is usually the simpler setting.

A whole branch of multi-agent learning investigates how an agent can coordinate and perform when interacting with teammates it has never seen before. Picture for example a robot playing soccer that must join an existing team it has never practiced with. This problem is usually referred to as *ad-hoc teamplay* or *ad-hoc coordination*. There exist different settings depending on, the extent of knowledge that the agent has about its potential teammates, and whether or not the learner is allowed to update its policy during the ad-hoc interactions to identify and fine-tune itself to the team (Stone et al., 2010; Barrett et al., 2011). More recent approaches usually forbid fine-tuning the ad-hoc teams from learning updates but still assume some degree of knowledge about teammates. For example, some works assume access to offline demonstrations of the teammates' behaviors that can be used to guide the agent's self-play training toward adopting the appropriate equilibrium (or "social conventions") of its future teammates (Lerer and Peysakhovich, 2019; Tucker et al., 2020). Similarly, Carroll et al. (2019) uses offline data to learn a model of the teammate's behavior and use it to train the agent to coordinate with that ally. On the other hand, Hu et al. (2020) consider a slightly different setting, Zero-Shot Coordination (see Figure 2.15), that does not require information about the specific behaviors of the potential teammates. Instead, ZSC aims at deriving agents that will perform well when teamed together even if they were not trained together. In other words, it assumes that all agents (teammates included) learn with the same algorithm.

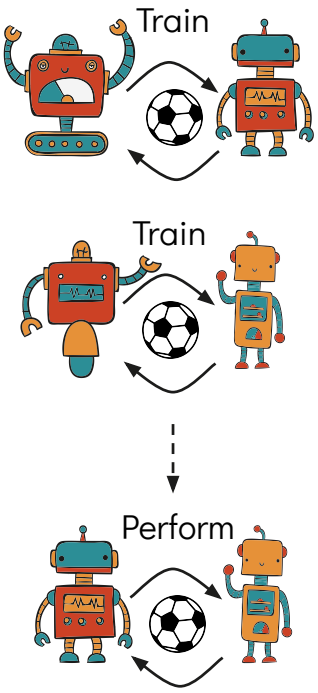


Figure 2.15: Zero-Shot Coordination. Agents that were trained independently must perform together.

2.3.3 Internal Models in Multi-Agent Learning

This subsection discusses the importance of internal models for successful multi-agent learning. Specifically, it investigates how these models can be used to simulate other agents or the environment in order to coordinate concurrent learners.

Modelling teammates

Internal models are mainly used by agents to represent their teammates and opponents present in the environment. They can then use these models to understand and predict how others are expected to behave so as to best adapt to them. Boutilier (1996a); Chalkiadakis and Boutilier (2003) propose to use Bayesian learning to estimate a distribution over other

agents' strategies in order to coordinate with them. Suryadi and Gmytrasiewicz (1999) follows a similar approach but instead model teammates' beliefs, capabilities, and preferences and derive how these drive an agent's behavior.

Agents modeling agents that are reciprocally modeling them in return brings infinite recursion: "agent 1 is going to do A because it assumes that agent 2 assumes that agent 1 assumes that...". Vidal and Durfee (1997) note that this cannot be feasibly sorted out and propose to categorize rational agents according to the complexity that they assume for their teammates. A level-0 agent considers that its teammates do not explicitly adapt their behavior to itself. A level-N agent assumes level-$(N-1)$ teammates, such that a level-1 agent considers that its teammates are level-0 agents and so on. Mundhe and Sen (2000a) explore level-0, level-1, and level-2 agents and concludes that in most domains level-1 agents are enough with some domains being already solved by level-0 agents. Foerster et al. (2018a) report similar findings and note that level-1 LOLA agents are immune to being manipulated by higher-level agents. While some works present tasks in which good performances can be achieved by level-0 learners (Mundhe and Sen, 2000a; Sen and Sekaran, 1998; Sen et al., 1994), most research advocates for level-1 learners. Banerjee et al. (2000); Sen et al. (2003) for instance showcase experiments in which level-1 learners can build a form of mutual trust by modeling each others' action probability distributions. Tambe (1995) notes the benefits of tracking the decisions of both groups of agents and individuals. Finally, learners can also benefit from modeling agents in purely competitive settings (Carmel and Markovitch, 1994; Uther and Veloso, 1997).

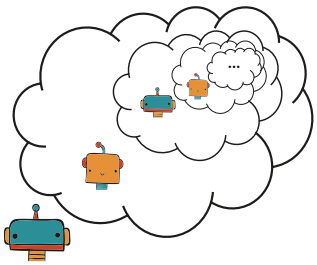


Figure 2.16: Infinite recursive thinking arises when agents reason about other agents that are reasoning about them.

Although modeling other agents can greatly benefit concurrent learners, some authors advise caution and note that the initial beliefs of an agent about its teammate can greatly influence the resulting global strategy (Hu and Wellman, 1996; Wellman and Hu, 1998). Sometimes, if an agent's assumption about its teammates' behavior is wrong, agent modeling can be misleading and prevent convergence to optimal behavior, yielding worse team performance than for level-0 agents. Following this, Hu and Wellman (1998), advocate for the conservative approach of minimizing an agent's assumption about the other agents' policies. This has led researchers to investigate agents that automatically assess if other agents are cooperating or competing in order to reciprocate adequately (Sekaran and Sen, 1995; Sen and Sekaran, 1996). Nowak and Sigmund (1998) distinguishes between two types of reciprocity: *direct* – agent 1 helps agent 2 and expects agent 2 to help in the future – and *indirect* – agent 1 helps agent 2 and expects to receive help from different agents. This latter form of reciprocity requires a "degree of acquaintanceship" that measures the likelihood of an agent knowing another's reputation. They argue that cooperation can occur if the degree of acquaintanceship is greater than the ratio between the cost of altruistic help and its benefits

to the recipient. Ito (1997) share similar considerations but do not rely on an explicit model of reputation. Instead, authors provide agents with a history of their partner's previous moves with other players.

Another, more recent, approach is to model others in order to simulate them in the environment. In Claus and Boutilier (1998), agents observe other agents' actions and learn a model of their current policy. They then use these models to simulate their teammates through *ficticious play* and estimate their own actions' Q-value in the context of other agents' current strategies. This approach is also common in the offline setting where a learner cannot directly interact with the other agents. Carroll et al. (2019) for instance leverage offline data to learn a model of the teammate's behavior and use it to train the agent so that it coordinates efficiently with that ally at test time.

Instead of reasoning about a partner's current behavior, some works try to infer a partner's updated strategy. For instance, Zhang and Lesser (2010) propose a MARL algorithm that leverages policy prediction to converge faster and more consistently to Nash equilibria in iterated games. The key idea is in essence similar to the later work of Foerster et al. (2018a): by predicting future policies, an agent can adjust its strategy to the forecasted behavior of other players instead of their current one. The main difference is that LOLA agents actively try to influence their opponents' future strategies.

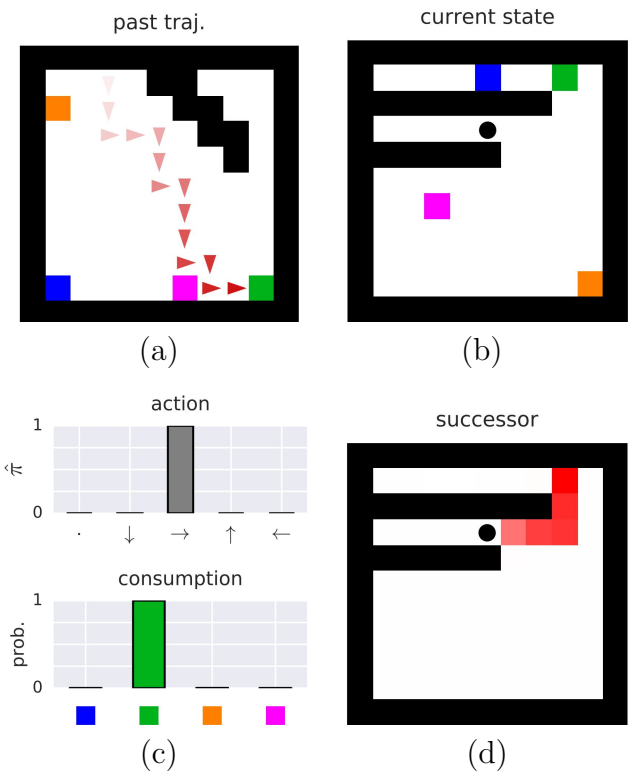


Figure 2.17: Model predictions excerpt from Rabinowitz et al. (2018): (a) the past trajectory of the agent given to the model, (b) the current state of the agent, (d) and (c) based on the past trajectory and the current state, the model predicts that the agent is going to go right and up in order to consume the green object.

Teammate modeling can also benefit agents even when their policy-optimizing algorithm does not explicitly leverage the prediction about other agents' behavior. Ndousse et al. (2021) shows that agents learn more rapidly, better, and more generalizable policies if they are trained with expert agents present in the environment. However, these benefits only hold if learners are equipped with the auxiliary task of modeling the expert agents by predicting their behavior. Yet, the agents' policy optimization scheme does not use the behavior prediction module in any way to adjust one's strategy.

A myriad of methods, (Lowe et al., 2017; Foerster et al., 2018a; Jaques et al., 2019), are based on the learning centralization assumption of Centralized Training and Decentralized Execution (CTDE), however, these works also experiment with relaxing this assumption and using teammate modeling instead. While the overall learning performance often drops because of the teammate-model inaccuracies, it remains a promising approach for more decentralized and scalable concurrent learning. This has motivated researchers to move away from naive behavioral cloning-based approaches in order to devise more elaborate and scalable ways of modeling others. Raileanu et al. (2018) propose an approach where a learner predicts other agents' behaviors by using its own policy. In settings where agents are similar to one another, this allows leveraging teammate modeling for free. Rabinowitz et al. (2018) propose to learn sophisticated models of others from a meta-learning approach. The same network is used to model a variety of agents and extract both *general behavior patterns* and *agent-specific features*, the latter being formed at test time from the current trajectory of the agent being modeled. This results in an elaborate agent-modeling module that discovers abstractions in the space of behaviors and is able to infer agents' belief states, goals, and goal-directed behaviors (see Figure 2.17 and Figure 2.18).

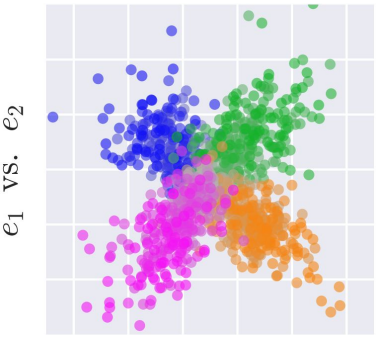


Figure 2.18: Model agent-embeddings excerpt from Rabinowitz et al. (2018): 2D behavior embeddings of the model, each dot represents a monitored agent and is colored by the latter's ground truth preferred object. The model has learned to identify and represent agents based on their preferences.

Modelling the world

Agents do not only model other agents, they can also learn models of the environment, sometimes referred to as world models, and leverage them for planning, learning abstractions, or generating synthetic experiences (Ha and Schmidhuber, 2018). While these ideas have been around for quite some time in the single agent setting (Craik, 1967; Dennett, 1975; Sutton, 1981, 1991), they have yet to be extensively investigated for concurrent multi-agent learning.

To improve sample efficiency in terms of interactions with the environment, several works (Chockalingam et al., 2018; Willemsen et al., 2021) have extended world-models approaches and single-agent model-based optimization (Janner et al., 2019) to the multi-agent setting by leveraging the Centralized Training and Decentralized Execution (CTDE) framework. Zhang et al. (2021) relax the centralization requirements by also building models of the opponents and using them to generate the synthetic training data in a more decentralized fashion.

In the cooperative setting, Egorov and Shpilman (2022) adds a communication module through which agents exchange their current internal states and derive a global team em-

bedding that can be used to coordinate their behaviors. Similarly, Pretorius et al. (2021) proposes agents that learn to coordinate by communicating encodings of their model rollouts and predicted outcomes. By exchanging these individually forecasted trajectories, agents can better asses teammates' beliefs and intentions and jointly plan a strategy. Since this communication process also occurs during training – and is optimized end-to-end during agents' concurrent learning – one can also see it as a mechanism to coordinate the agents' learning.

Sessa et al. (2022) present a sample efficient algorithm that balances exploration and exploitation to collect data and iteratively build a world model throughout the agents' learning. At each iteration, the current world model is used to optimistically construct a simulated game that approximates the agents' task and a subroutine derives the corresponding equilibrium policies. Authors present theoretical bounds on the dynamic regret of the agents – that is the payoff they miss by using a sub-optimal policy while learning – and on the convergence rate towards the approximate equilibria of the true underlying Markov Game. This approach is reminiscent of the early work of Chalkiadakis and Boutilier (2003) which investigates the generalized *exploration-exploitation* tradeoff that arises in MARL. A compromise has to be found between learning the equilibrium and exploring the strategy space to discover more desirable team equilibria. The authors propose to learn bayesian models for both the environment and the other agents. Planning is done by assessing the value of the actions under the current models – for example by its potential to influence other agents – but also for the *value of information* it provides. This latter relates to the ability of an action to provide information that might change future decisions, for instance by gaining knowledge about other agents' strategies or the world's dynamics.

2.3.4 Shared Incentives in Multi-Agent Learning

This subsection considers two types of incentives that agents can share to help them coordinate: objectives – that define *what* the agents should learn – and conventions – that specify *how* they should learn.

Shared objectives

Here we focus on the importance of rewards to define objectives and review the specific case of the emergence of communication.

The importance of shared rewards. The role of shared goals and rewards to promote cooperation and coordination has already been mentioned with respect to credit assignment in Subsection 2.1.2. Indeed, in the absence of shared rewards, individually motivated agents might end up locked in competitive strategies regardless of cooperation being the better option. Early on, Mataric (1994a) have advocated for small *vicarious reinforcements* where an agent directly gets portions of the rewards received by agents around it. This local sharing

of the rewards allows for spreading individual rewards to the whole team and can regroup it towards a common objective, eventually fostering cooperation.

This is a common idea in the literature and many variations of it have been investigated. Peysakhovich and Lerer (2017) show that globally sharing individual rewards in the Stag Hunt coordination problem leads agents to converge to the more risky but higher payoff Nash equilibrium. However, completely sharing rewards might impede learning in more complex environments by interfering with credit assignments. Indeed, an agent cannot differentiate anymore if it is responsible for the reward it receives or if it comes from the other agent's deed. This can lead to "lazy agent" learning pathologies. A prevalent alternative is therefore to use a scheduled learning curriculum to adapt the reward from selfish and individual local rewards at the beginning of training to cooperative shared global rewards toward the end of it. OpenAI et al. (2019) leverage such progressive credit assignment schedule with their *team spirit* reward to ease credit assignment in the early learning phases of the complex Dota 2 video game. Similarly, Lerer and Peysakhovich (2017) highlight the challenges of cooperating – which they define as maximizing the joint payoff – in social dilemmas where individuals are tempted to increase their individual outcomes regardless of whether or not this would hurt the global payoff. The authors propose a variation on self-play based on a two-phase reward curriculum where agents start with selfish individual rewards to learn the game mechanics and finish with shared rewards to promote cooperation. Sharing rewards is a basic, yet efficient way of aligning agents' objectives and having them coordinate.

Works have come up with more sophisticated ways of aligning agents' goals. Ahilan and Dayan (2019) for instance considers a hierarchical learning decomposition in which a manager agent sets the goals and rewards of other agents to organize and coordinate their learning. A more horizontal and distributed version of this approach is proposed by Yang et al. (2020) where agents influence each other by actively rewarding each other so as to coordinate toward a common goal.

The emergence of communication. The importance of aligning the agents' objectives has been extensively discussed by the research community focusing on the emergence of communication. Rightly, effective and efficient communication is both practical to measure and a good indicator of agents' cooperation and coordination.

In Lazaridou et al. (2016)'s experiments, agents share the same rewards in order to coordinate and emerge a communication protocol that allows them to solve a referential game. Cao et al. (2018) presents an environment where agents that have different preferences over items must negotiate and agree on which agent gets which items. Agents interact by sending proposals about which items it would take provided that the other agent accepts the offer. In addition to the proposals, agents can use cheap talk, that is sending arbitrary messages that do not impact the game transitions or rewards directly. Results show that, while selfish agents can negotiate and fairly split the items, they do this suboptimally and do not leverage cheap talk. On the other hand, prosocial agents that share a joint reward

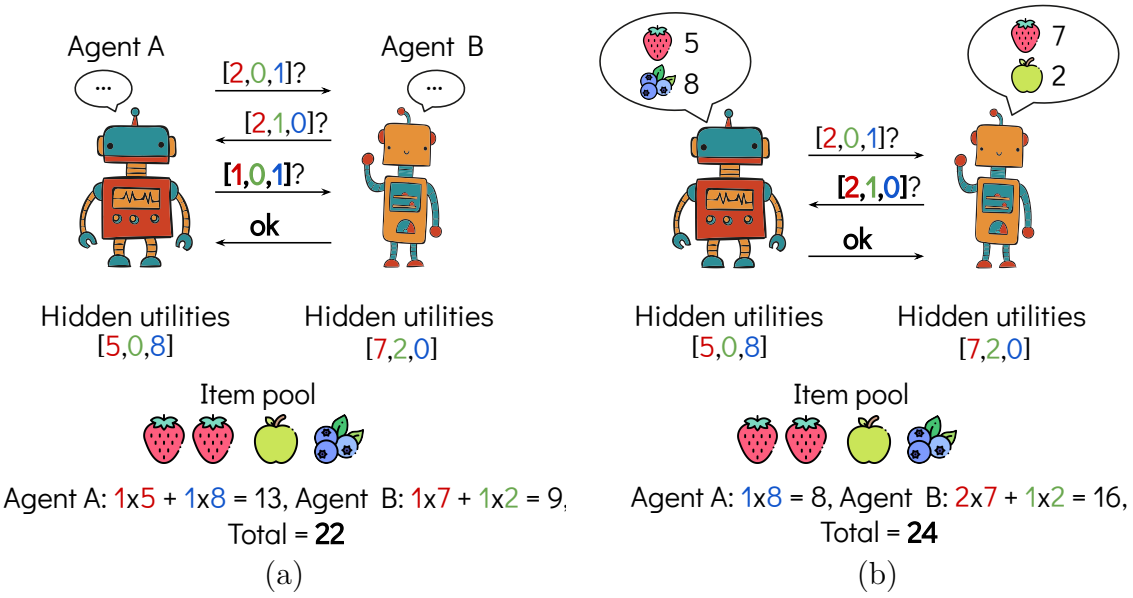


Figure 2.19: Negotiation game of Cao et al. (2018). Agents have different preferences over items must split the item pool between them. They interact by sending proposals of the items they would like to have and they can also communicate arbitrary messages for free. (a) Selfish agents do not learn to communicate and split items suboptimally (the total outcome could be higher). (b) Agents that share rewards learn to communicate their hidden preferences and are able to split items so as to maximize the total outcome.

learn to use the messages to communicate their preferences and split the items optimally (see Figure 2.19). These results highlight the importance of aligned goals – and thus cooperation– for the emergence of communication with cheap talk. It presents experimental support for the theoretical results of Crawford and Sobel (1982); Nowak and Krakauer (1999).

A more in-depth discussion on the relationship between cooperation and communication is proposed by Noukhovitch et al. (2021). First, they experimentally retrieve Crawford and Sobel (1982)'s result that communication is proportional to cooperation. Authors thus show that communication with cheap talk can emerge in partially competitive scenarios provided that both agents individually benefit from it. If one agent can dominate without communication it will shut down and prevent its development, despite the other agent actively trying to make contact. Additionally, in order to unify the cooperative and competitive settings with regard to communication, the authors introduce the notion of information transfer and differentiate between communication and manipulation (Barrett and Skyrms, 2017; Dawkins and Krebs, 1978; Hinde, 1981). In this new taxonomy, information transfer is defined as the amount of knowledge that passes – or spills – from the sender to the receiver, regardless of which agent benefits from it. Communication considers that both agents benefit from the information transfer while manipulation profits the sender but not the receiver.

This section highlights the importance of shared objectives – mainly through global rewards – in aligning agents learning and fostering cooperation and coordination, be it in the physical or communicative space. In addition to aligning agents by sharing rewards, some works emphasize the importance of supplementary shared objectives. Liu et al. (2021) for instance presents a MARL method in which it is crucial that agents share goals to promote an efficient and coordinated cooperative exploration.

Shared conventions

In addition to sharing objectives, agents may share conventions. Such *social norms* structure interactions, coordinate learning, and enable teams to converge to common strategies.

Conventions to structure learning. Jennings (1996) argues that all coordination mechanisms can ultimately be reduced to *commitments* and their associated (social) *conventions*. Commitments are covenants to follow specific courses of action, they ensure a degree of predictability and allow agents to plan while taking others into consideration. Conventions are best seen as procedures on how to monitor and adapt commitments to changing contexts. They provide flexibility and define the mechanisms to (1) asses if current commitments are valid given the actual circumstances, and (2) reassess and adapt commitments.

Jennings (1996)'s focus is more on procedural agents and how to coordinate their planning and execution, yet conventions are also present in learning agents. In that case, conventions tend to structure learning by introducing biases and prior knowledge in order to nudge and unify the learned behavior. As we have pointed out in Subsection 2.1.5, "rational agents" tend to converge to suboptimal Nash equilibria rather than to globally team-optimal solutions (Lichbach, 1996). On the other hand, many works (Lauer, 2000; Claus and Boutilier, 1998; Kapetanakis and Kudenko, 2002a; Panait et al., 2003) have pointed out that agents that are optimistic about their partners' performance and cooperativeness tend to converge to better team strategies than strictly rational ones. They argue that in the context of cooperation, agents' rationality should come after the requirements for optimal team strategy. For instance, it feels absurd to enforce rationality in the Prisoner's dilemma as it prescribes sub-optimal solutions that human participants would never choose. Instead, artificial agents should follow human intuition and cooperate[1]. Optimistic learning can be implemented in many ways, Lauer (2000) proposes to update an agent's Q-values by estimating the best possible cooperative behavior of other agents to its action. Similarly, in the offline RL setting, Jiang and Lu (2021) artificially makes high-value states more likely by assuming that the whole team will strive to reach these promising situations. The authors argue that high-value states are underrepresented in offline non-expert data because the mistake of one agent can

[1]The question of what makes humans irrational and prevents them from exploiting benevolent teammates is an interesting one and might relate to altruism, social pressure, mores, reputation, fear of retribution, etc.

deprive the whole team from a good outcome: in suboptimal teams, miscoordination is more likely than coordination.

Conventions to coordinate on common strategies. Conventions might also enable teams to coordinate by specifying how to deal with symmetry. Many situations where multiple optimal equilibria exist require agents to arbitrarily break ties in order to converge to the same strategy (Shoham et al., 2004). Boutilier (1996b) presents such a simple coordination game where each agent has to choose between *left* and *right* options but only gets rewarded if the other agent picks the same option. The authors propose guidelines to design conventions and social laws based on a lexicographic ordering of actions. Prioritizing some action over another effectively break ties during strategy selection. However, such conventions are not learned and lexicographic orderings are most often ungrounded and arbitrary. This prevents generalization to unacquainted agents. Hu et al. (2020) proposes Other-play, a variation on self-play that is invariant to the inherent symmetries of the environment. The method forces agents to learn coordinating conventions that are grounded and can generalize at test-time to teams that have not been trained together. Lerer and Peysakhovich (2019) investigates how to identify and adapt to existing conventions. They leverage imitation learning on offline data to align the learned MARL behaviors to the social conventions in use. That way agents can blend in and coordinate well when they enter an existing group.

Emergence of linguistic conventions Finally, the field of experimental semiotics, which studies the novel forms of communication that agents develop when they cannot use preexisting ones, abounds with works investigating the emergence of social conventions in populations of agents and across generations (Galantucci and Garrod, 2011). While most of these works focus on human learning (Galantucci and Steels, 2008; Vollmer et al., 2014), there is an important effort to bring these ideas and methods to the AI community with a focus on the emergence of language (Beuls and Steels, 2013; Spranger and Steels, 2015; Steels, 2011; Steels and Hild, 2012; Steels and Szathmáry, 2018) and its acquisition for understanding feedback and following instruction (Lopes et al., 2011; Cederborg and Oudeyer, 2014; Grizou et al., 2013, 2014b,a).

Chapter 3

Background

This chapter is not intended to be a comprehensive or in-depth course on the presented notions. Instead, it aims at giving the required background to understand the methods and derivations exposed in this manuscript. The reader is referred to Littman (1994); Oliehoek and Amato (2015); Sutton and Barto (2018); Levine et al. (2020) for more extensive material.

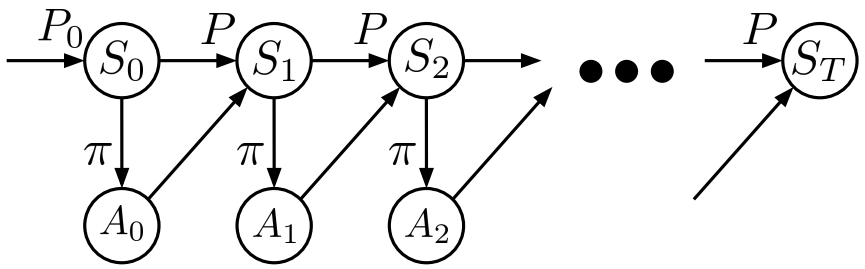
3.1 Modelling Sequential Decision Processes

This section covers the prevalent formalisms for single and multi-agent sequential decisions making.

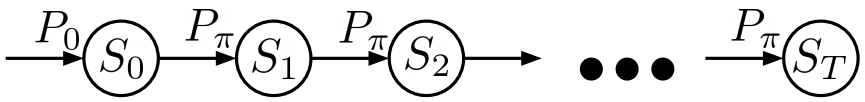
3.1.1 Markov Decision Processes (MDPs)

MDPs provide a mathematical framework to handle decision-making in situations where outcomes are stochastic and only partially under the decision maker's control (Puterman, 2014). We consider the discrete-time stochastic control process modeled by the T-horizon γ-discounted MDP $\langle \mathcal{S}, \mathcal{A}, P, P_0, \gamma, r, T \rangle$. For simplicity, we assume that $\mathcal{S}$ and $\mathcal{A}$, respectively the set of all possible states and actions, are finite. Successor states follow the state-action transition distribution $P(s'|s,a) \triangleq \Pr\{S' = s' | S = s, A = a\} \in [0,1] \ \forall \ s' \in \mathcal{S}, \ s \in \mathcal{S}, \ a \in \mathcal{A}$ with S', S, A respectively the next state, current state, and current action random variables. The initial state random variable S_0 is drawn from the initial state distribution $P_0(s) \triangleq \Pr\{S_0 = s\} \in [0,1] \ \forall s \in \mathcal{S}$. Transitions are rewarded with $r(s,a,s') \in \mathbb{R}$ with r being bounded and we abuse notation by defining $r(s,a) = \mathbb{E}_P[r(s,a,s')]$ where the expectation over next states is taken with respect to the transition distribution. The discount factor and the episode horizon are $\gamma \in [0,1]$ and $T \in \mathbb{N} \cup \{\infty\}$, where $T < \infty$ for $\gamma = 1$. Finally, we consider stationary stochastic policies $\pi : \mathcal{S} \times \mathcal{A} \to]0,1[$ such that $\pi(a|s) \triangleq \Pr\{A = a | S = s\}$.

As shown in Figure 3.1 (a), a policy π and a MDP form a Probabilistic Graphical Model (PGM) with the state and action at time t as sequential random variables (respectively



(a) The Probabilistic Graphical Model view



(b) The Markov chain view

Figure 3.1: PGM illustration. A policy and a MDP form a Markov chain.

S_t and A_t). Importantly, note that the next state only depends on the current state and action. Since the current action also only depends on the current state we can define the policy-induced state transition function:

$$\begin{aligned} P_\pi(s'|s) &\triangleq \Pr\{S' = s'|S = s\} \\ &= \sum_a \Pr\{A = a|S = s\}\Pr\{S' = s'|S = s, A = a\} \\ &= \sum_a \pi(a|s)P(s'|s,a) \\ &= \mathbb{E}_\pi[P(s'|s,a)], \end{aligned} \tag{3.1}$$

where $\mathbb{E}_\pi[\cdot]$ denotes the expected value of a random variable (or of a function of random variables) under the distribution π (i.e., $A \sim \pi$). With this new transition function, we see that the PGM is actually a Markov chain that respects the Markov property: the next state only depends on the current one (see Figure 3.1(b)).

Executing the policy on the MDP produces realizations that we call trajectories, or rollouts:

$$\begin{aligned} \tau &= (S_0 = s_0, A_0 = a_0, S_1 = s_1, A_1 = a_1, ..., S_{T-1} = s_{T-1}, A_{t-1} = a_{T-1}, S_T = s_T) \\ &= (S = s)_{\{0:T\}}, (A = a)_{\{0:T-1\}}. \end{aligned} \tag{3.2}$$

The probability of trajectory τ under policy π is

$$\begin{aligned} p_\pi(\tau) &\triangleq \Pr\{(S=s)_{\{0:T\}}, (A=a)_{\{0:T-1\}}\} \\ &= \Pr\{S_0 = s_0\} \prod_{t=0}^{T-1} \Pr\{A = a_t | S = s_t\} \Pr\{S' = s_{t+1} | S = s_t, A = a_t\} \\ &= P_0(s_0) \prod_{t=0}^{T-1} \pi(a_t|s_t) P(s_{t+1}|s_t, a_t). \end{aligned} \tag{3.3}$$

For $0 \leq t < T$ we can define the following marginals:

$$\begin{aligned} d_{t,\pi}(s) &\triangleq \Pr\{S_t = s\} \\ &= \sum_{\{\tau | S_t = s\}} p_\pi(\tau) \\ &= \sum_{x_{\{0:T\}\setminus\{t\}}} \sum_{y_{\{0:T-1\}}} \Pr\{(S=x)_{\{0:T\}\setminus\{t\}}, (A=y)_{\{0:T-1\}}, S_t = s\}, \end{aligned} \tag{3.4}$$

$$\begin{aligned} d_{t,\pi}(s,a) &\triangleq \Pr\{S_t = s, A_t = a\} \\ &= \Pr\{A_t = a | S_t = s\} \Pr\{S_t = s\} \\ &= \pi(a|s) d_{t,\pi}(s), \end{aligned} \tag{3.5}$$

where $\{\tau | S_t = s\}$ is the set of all trajectories such that the state at time t is equal to s and $\sum_{s_{\{0:T\}\setminus\{t\}}} \sum_{a_{\{0:T-1\}}}$ indicates that we sum over all the possible values of all the variables $s_0, s_1, ..., s_{t-1}, s_{t+1}, .., s_T$ and $a_0, a_1, ..., a_{T-1}$. This implies T summations over $\mathcal{S}$, and T summations over $\mathcal{A}$. With these marginals, we define the normalized discounted state (and state-action) occupancy measures as

$$d_\pi(s) \triangleq \frac{1}{Z(\gamma, T)} \sum_{t=0}^{T-1} \gamma^t d_{t,\pi}(s), \tag{3.6}$$

$$d_\pi(s,a) \triangleq \frac{1}{Z(\gamma, T)} \sum_{t=0}^{T-1} \gamma^t d_{t,\pi}(s,a) = d_\pi(s)\pi(a|s), \tag{3.7}$$

where the partition function $Z(\gamma, T)$ is equal to $\sum_{t=0}^{T-1} \gamma^t$. Intuitively, the state (or state-action) occupancy measure can be interpreted as the discounted visitation distribution of the states (or state-action pairs) that the agent encounters when navigating the MDP with policy π.

The sum of the discounted rewards along a trajectory is called the return and is expressed

as:

$$G(\tau) = \sum_{t=0}^{T-1} \gamma^t \, r(s_t, a_t). \tag{3.8}$$

For a given MDP, the expected return of a policy can be expressed in terms of the occupancy measures as follows:

$$J_\pi[r(s,a)] \triangleq \mathbb{E}_{p_\pi}\left[G(\tau)\right] = Z(\gamma, T)\, \mathbb{E}_{d_\pi}[r(s,a)]. \tag{3.9}$$

Or expressed in terms of the state action marginals:

$$J_\pi[r(s,a)] = \sum_s \sum_a r(s,a) \sum_{t=0}^{T-1} \gamma^t d_{t,\pi}(s,a) = \sum_{t=0}^{T-1} \gamma^t \mathbb{E}_{d_{t,\pi}}[r(s,a)]. \tag{3.10}$$

See derivations on the next page where we use $T' = T - 1$ and remove the final terminal state of trajectories to ease notations.

$$
\begin{aligned}
J_\pi[r(s,a)] &\triangleq \mathbb{E}_{p_\pi}\left[G(\tau)\right] \\
&= \underbrace{\sum_{\tau} p_\pi(\tau) G(\tau)}_{\text{expand with Eqs. (3.2) and (3.8)}} \\
&= \underbrace{\sum_{x_{\{0:T'\}}} \sum_{y_{\{0:T'\}}}}_{\text{factorize independent terms out of sums}} \Pr\{(S=x, A=y)_{\{0:T'\}}\} \underbrace{\sum_{t=0}^{T'} \gamma^t}\ r(x_t, y_t) \\
&= \sum_{t=0}^{T'} \gamma^t \underbrace{\sum_{x_{\{0:T'\}}} \sum_{y_{\{0:T'\}}}}_{\text{split sums to make } t \text{ index explicit}} \Pr\{(S=x, A=y)_{\{0:T'\}}\} r(x_t, y_t) \\
&= \sum_{t=0}^{T'} \gamma^t \underbrace{\sum_{x_t} \sum_{y_t} \sum_{x_{\{0:T'\}\setminus\{t\}}} \sum_{y_{\{0:T'\}\setminus\{t\}}} \Pr\{(S=x, A=y)_{\{0:T'\}\setminus\{t\}}, S_t=x_t, A_t=y_t\} r(x_t, y_t)}_{\text{for clarity, replace } x_t, y_t \text{ with } s, a} \\
&= \sum_{t=0}^{T'} \gamma^t \sum_{s} \sum_{a} \underbrace{\sum_{x_{\{0:T'\}\setminus\{t\}}} \sum_{y_{\{0:T'\}\setminus\{t\}}}}_{\text{factorize independent terms out of sums}} \Pr\{(S=x, A=y)_{\{0:T'\}\setminus\{t\}}, S_t=s, A_t=a\} \underbrace{r(s,a)} \\
&= \sum_{t=0}^{T'} \gamma^t \sum_{s} \sum_{a} r(s,a) \underbrace{\sum_{x_{\{0:T'\}\setminus\{t\}}} \sum_{y_{\{0:T'\}\setminus\{t\}}} \Pr\{(S=x, A=y)_{\{0:T'\}\setminus\{t\}}, S_t=s, A_t=a\}}_{\text{law of total probability}} \\
&= \sum_{t=0}^{T'} \gamma^t \sum_{s} \sum_{a} r(s,a) \underbrace{\Pr\{S_t=s, A_t=a\}}_{\text{Eq. (3.5)}} \\
&= \underbrace{\sum_{t=0}^{T'} \gamma^t \sum_{s} \sum_{a} r(s,a)}_{\text{shift order of independent sums}} d_{t,\pi}(s,a) = \sum_{t=0}^{T'} \gamma^t \mathbb{E}_{d_{t,\pi}}[r(s,a)] \\
&= \sum_{s} \sum_{a} r(s,a) \underbrace{\sum_{t=0}^{T'} \gamma^t d_{t,\pi}(s,a)}_{\text{Eq. (3.7)}} \\
&= Z(\gamma, T) \sum_{s} \sum_{a} r(s,a) d_\pi(s,a) \\
&= Z(\gamma, T)\, \mathbb{E}_{d_\pi}[r(s,a)].
\end{aligned}
$$

3.1.2 Partially Observable Markov Decision Processes (POMDPs)

POMDPs are a generalization of MDPs in which the agent does not have access to the true state of the environment s but instead observes it through a stochastic observation function $\mathcal{O}$ such that $o \sim \mathcal{O}(o|s)$. The agent must therefore rely on action-observation histories, defined as $h_t = \{o_0, a_0, ..., o_t\} \in \mathcal{H}$, to take decisions and $a_t = \pi(a|h_t)$. Learning in a POMDP is much more challenging than learning in a MDP and most theoretical results do not hold. In practice, if the true state s_t can be inferred from the history h_t, then learning is possible.

3.1.3 Decentralized Markov Decision Processs (Dec-MDPs) and Decentralized Partially Observable Markov Decision Processes (Dec-POMDPs)

Dec-POMDPs are the multi-agent extension of POMDPs (Bernstein et al., 2002; Nair et al., 2003). A Dec-POMDP of N agents is defined by the tuple $\langle \mathcal{S}, P, P_0, \gamma, T, \{O^i, \mathcal{A}^i, r^i\}_{i=1}^N \rangle$ where $\mathcal{S}$, P, P_0, γ, and T are respectively the set of all possible states, the state-action transition distribution, the initial state distribution, the discount factor and the horizon. While these are global properties of the environment, O^i, $\mathcal{A}^i$, and r^i are individually defined for each agent i. They are respectively the observation function, the set of all possible actions, and the reward function. At each time-step t, the global state of the environment is given by $s_t \in \mathcal{S}$ and every agent's individual action vector is denoted by $a_t^i \in \mathcal{A}^i$. At each time-step, each agent takes an action using its policy $\pi^i : \mathcal{H}^i \times \mathcal{A}^i \rightarrow]0, 1[$ and its own action-observation history $h_t^i = \{o_0^i, a_0^i, ..., o_t^i\} \in \mathcal{H}^i$ such that $o_t^i \sim O^i(o^i|s_t) \in \mathcal{O}^i$ and $a_t^i \sim \pi^i(a^i|h_t^i) \in \mathcal{A}^i$. The initial state s_0 is sampled from the initial state distribution $P_0 : \mathcal{S} \rightarrow [0, 1]$ and the next state s_{t+1} is sampled from the probability distribution over the possible next states given by the transition function $P : \mathcal{S} \times \mathcal{S} \times \mathcal{A}^1 \times ... \times \mathcal{A}^N \rightarrow [0, 1]$. Finally, at each time-step, each agent receives an individual scalar reward r_t^i from its reward function $r^i : \mathcal{S} \times \mathcal{S} \times \mathcal{A}^1 \times ... \times \mathcal{A}^N \rightarrow \mathbb{R}$. We will use bold to refer to joint variables, e.g. $\boldsymbol{a}_t \triangleq [a_t^1, a_t^2, ..., a_t^N]$ and $\boldsymbol{\pi} \triangleq [\pi^1, ..., \pi^N]$.

Since rewards and transitions depend on all of the agents, each agent's return also depends on the other agents:

$$J_{\boldsymbol{\pi}}[r^i(s, \boldsymbol{a})] = \mathbb{E}_{p_{\boldsymbol{\pi}}}[\sum_{t=0}^{T-1} \gamma^t r^i(s_t, \boldsymbol{a}_t)] \tag{3.11}$$

Where $p_{\boldsymbol{\pi}}$ indicates the trajectory distribution when actions are sampled according to $\boldsymbol{\pi}$, i.e., each agent uses its policy π^i.

Agents act independently and the joint policy decomposes into individual independent policies:

$$\boldsymbol{a}_t \sim \boldsymbol{\pi}(\boldsymbol{a}|s_t) \triangleq \prod_{i=0}^{N} \pi^i(a^i|h_t^i). \tag{3.12}$$

Centralized Training and Decentralized Execution (CTDE)

In the CTDE framework (Lowe et al., 2017; Foerster et al., 2018b), individual observations, policies, and value functions are available to all the agents at training time. For simplicity, most works also assume access to the global state s_t and the observation functions O^i. When this is unfeasible, the global state can be defined as the concatenation of the agents' observations, i.e., $s_t = \{o_t^1, ..., o_t^N\}$. Dec-MDPs are special cases of Dec-POMDPs in which such concatenation of observations fully observes the environment, i.e., the true global state can be inferred from it.

Fully cooperative case

In fully cooperative tasks, all the agents receive the same reward and

$$r_t^i = r_t = r(\boldsymbol{s}_t, \boldsymbol{a}_t) \in \mathbb{R} \, \forall i, t. \tag{3.13}$$

The agents' shared objective is therefore to maximize the expected team return $J_{\boldsymbol{\pi}} = \mathbb{E}_\tau \left[G(\tau) \right]$ where $G(\tau) = \sum_t \gamma^t r_t$ with discount factor γ, $\tau = \{\boldsymbol{s}_0, \boldsymbol{a}_0, r_0, ...\boldsymbol{s}_T\}$ is a trajectory with absorbing state $\boldsymbol{s}_T$, $\boldsymbol{s}_{t+1} \sim P(\boldsymbol{s}'|\boldsymbol{s}_t, \boldsymbol{a}_t)$, and $\boldsymbol{a}_t \sim \boldsymbol{\pi}$.

3.2 Single- and Multi-Agent Reinforcement Learning

This section covers the Reinforcement Learning based approaches to single and multi-agent learning.

3.2.1 Single-Agent Reinforcement Learning (RL)

The field of RL aims at finding the optimal policy π^*, i.e., the one that maximizes the expected return (Sutton and Barto, 2018):

$$\pi^* = \arg\max_\pi J_\pi[r(s, a)]. \tag{3.14}$$

It can be shown that, for any MDP, there exists an optimal deterministic policy π^* such that:

$$\pi^*(s) = \arg\max_a Q^*(s, a), \tag{3.15}$$

$$Q^*(s, a) = r(s, a) + \gamma \sum_{s'} P(s'|s, a) V^*(s'), \tag{3.16}$$

$$V^*(s) = \max_a Q^*(s, a), \tag{3.17}$$

where $V^*(s)$ and $Q^*(s,a)$ are respectively the optimal state value and the optimal state-action value. These can be computed using Value Iteration (Bellman, 1966), a Dynamic Programming algorithm that iterates the following update rule to convergence:

$$Q^{k+1}(s,a) \leftarrow r(s,a) + \gamma \sum_{s'} P(s'|s,a)V^k(s') \quad \text{with} \quad Q^0(s,a) = r(s,a). \tag{3.18}$$

Maximum Entropy (MaxEnt) Reinforcement Learning

In the entropy-regularized Reinforcement Learning framework (Haarnoja et al., 2018), the optimal soft-policy maximizes its entropy at each visited state (in addition to the standard RL objective):

$$\pi^* \triangleq \arg\max_{\pi} J_\pi[r(s,a) + \alpha\mathcal{H}(\pi(\cdot|s))], \quad \mathcal{H}(\pi(\cdot|s)) \triangleq \mathbb{E}_\pi[-\log(\pi(a|s))].$$

As shown in Ziebart (2010) and Haarnoja et al. (2017), the corresponding optimal soft-policy is

$$\pi^*_{\text{soft}}(a|s) = \exp\left(\alpha^{-1}\, A^*_{\text{soft}}(s,a)\right) \quad \text{with} \quad A^*_{\text{soft}}(s,a) \triangleq Q^*_{\text{soft}}(s,a) - V^*_{\text{soft}}(s), \tag{3.19}$$

$$V^*_{\text{soft}}(s) = \alpha \log \sum_a \exp\left(\alpha^{-1}\, Q^*_{\text{soft}}(s,a)\right), \; Q^*_{\text{soft}}(s,a) = r(s,a) + \gamma \mathbb{E}_{s' \sim \mathcal{P}(\cdot|s,a)}\left[V^*_{\text{soft}}(s')\right]. \tag{3.20}$$

Policy Invariance

Ng et al. (1999) noted an interesting property of optimal policies: they are invariant under specific reward transformations. Specifically, if a policy π^* is optimal in a given MDP with reward $r(s,a,s')$, then it is also optimal with reward $\hat{r}(s,a,s')$ provided that it exists a real-valued function $\Phi : \mathcal{S} \to \mathbb{R}$ such that

$$\hat{r}(s,a,s') = r(s,a,s') + \gamma\Phi(s') - \Phi(s). \tag{3.21}$$

In which case $\hat{r}$ is said to be a potential-based reward shaping of r.

On-policy vs. Off-policy methods

An important distinction between RL methods revolves around the nature of interactions they can use to learn. *On-policy* methods require the interactions to be collected with the current policy, they include SARSA (Rummery and Niranjan, 1994; Sutton and Barto, 2018) and PPO (Schulman et al., 2017), and are often robust in the sense that they display stable learning curves and require less hyperparameter tuning. On the other hand, *off-policy* methods can learn from transitions collected with any policy and they often leverage

past interactions by using replay buffers in which they store collected interactions. Off-policy methods do not require collecting new interactions every time the policy is updated, therefore they usually use fewer environment interactions, a property referred to as *sample efficiency*. Q-learning (Watkins and Dayan, 1992; Sutton and Barto, 2018) and DDPG (Silver et al., 2014; Lillicrap et al., 2015) are example of off-policy methods.

3.2.2 Multi-Agent Reinforcement Learning (MARL)

In MARL each agent aims at maximizing its own expected return:

$$\pi^{i*} = \arg\max_{\pi^i} J_{\boldsymbol{\pi}}[r^i(s, \boldsymbol{a})] = \arg\max_{\pi^i} \mathbb{E}_{p_{\boldsymbol{\pi}}}[\sum_{t=0}^{T-1} \gamma^t r^i(s_t, \boldsymbol{a}_t)] \tag{3.22}$$

This optimization is more challenging than its single agent counterpart Eq. (3.14) because the return of an agent depends on quantities that it does not control or know of – such as the global state and the actions of the other agents. Moreover, since other agents change how they select their actions as they learn, the environment as observed by an agent is non-stationary, i.e., the transitions distributions and reward functions change over the course of training.

Multi-Agent Deep Deterministic Policy Gradient (MADDPG)

With MADDPG, the multi-agent extension of DDPG (Lillicrap et al., 2015), Lowe et al. (2017) proposed a centralized training procedure that makes the environment stationary during learning while allowing for decentralized policies. The term centralized is used when an agent has access to the observations and actions of all the agents, whereas if agents only access their own observations and actions we call it decentralized. In MADDPG, each agent i possesses its own deterministic policy π^i for action selection and critic Q^i for state-action value estimation, which are respectively parametrized by θ^i and ϕ^i. All parametric models are trained from collected transitions $\zeta_t \triangleq (\mathbf{o}_t, \mathbf{a}_t, \mathbf{r}_t, \mathbf{o}_{t+1})$ uniformly sampled from a replay buffer $\mathcal{D}$ (which makes it off-policy). Each centralized critic is trained to estimate the expected return for a particular agent i by minimizing the Q-learning loss (Watkins and Dayan, 1992):

$$\begin{aligned} L^i(\phi^i) &= \mathbb{E}_{\zeta_t \sim \mathcal{D}}\left[\frac{1}{2}\left(Q^i(\mathbf{o}_t, \mathbf{a}_t; \phi^i) - y_t^i\right)^2\right], \\ y_t^i &= r_t^i + \gamma Q^i(\mathbf{o}_{t+1}, \mathbf{a}_{t+1}; \bar{\phi}^i)\Big|_{a_{t+1}^j = \pi^j(o_{t+1}^j; \bar{\theta}^j)\,\forall j}. \end{aligned} \tag{3.23}$$

For a given set of weights w, we define its target counterpart $\bar{w}$, updated from $\bar{w} \leftarrow cw + (1-c)\bar{w}$ where c is a hyperparameter. Each policy is updated to maximize the expected

return of the corresponding agent i, yielding the Policy Gradient (PG) objective :

$$J^i_{PG}(\theta^i) = \mathbb{E}_{\mathbf{o}_t \sim \mathcal{D}} \left[Q^i(\mathbf{o}_t, \mathbf{a}_t) \Big|_{\substack{a^i_t = \pi^i(o^i_t; \theta^i), \\ a^j_t = \pi^j(o^j_t; \bar{\theta}^j)\, \forall j \neq i}} \right]. \tag{3.24}$$

Despite the multi-agent setting, agents are trained in a centralized and stationary environment since the value functions take into account *all* the agents' observation-action pairs when guiding an agent's policy. This mechanism can allow one to learn coordinated strategies that can then be deployed in a decentralized way.

Multi-Agent Proximal Policy Optimization (MAPPO)

MAPPO (Yu et al., 2022) is an extension of Proximal Policy Optimization (PPO) (Schulman et al., 2017) to the multi-agent setting. PPO is a Policy Gradient method that aims at maximizing the following objective:

$$J_{PG}(\theta) = \mathbb{E}_\pi[\log \pi_\theta(a_t|s_t) A_t]. \tag{3.25}$$

A_t is an estimation of the advantage function and $\mathbb{E}_\pi$ means that the expectation is taken with respect to transitions collected with the current π (which makes PPO an on-policy method). The above objective increases the likelihood of actions that yield a high advantage and reduces the likelihood of actions with a small advantage. Advantages are estimated using the Generalized Advantage Estimator (GAE):

$$A_t = \delta_t + (\gamma\lambda)\delta_{t+1} + ... + (\gamma\lambda)^{T-t+1}\delta_{T-1}, \tag{3.26}$$

$$\delta_t = r_t + \gamma V(s_{t+1}) - V(s_t). \tag{3.27}$$

where λ is a hyperparameter.

The value V is learned by minimizing:

$$L_V(\theta) = -\mathbb{E}_\pi \left[\frac{1}{2} (V_\theta(s_t) - G_t)^2 \right], \qquad G_t = \sum_{i=0}^{T-t-1} \gamma^i r_{t+i}. \tag{3.28}$$

PPO constrains the updates such that updated models remain in regions for which the collected transitions used to compute the objectives can still be trusted. For transitions collected with $\pi_{\theta,\text{old}}$ and clipping parameter ϵ:

$$J^{PPO}_{PG}(\theta) = \mathbb{E}_{\pi_{\theta,\text{old}}} \left[\min \left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta,\text{old}}(a_t|s_t)} A_t, \text{clip} \left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta,\text{old}}(a_t|s_t)}, 1-\epsilon, 1+\epsilon \right) A_t \right) \right]. \tag{3.29}$$

Similarly, for the value loss:

$$L_V^{PPO}(\theta) = \frac{1}{2}\mathbb{E}_{\pi_{\theta,\text{old}}}\left[\max\left((V^{\text{clip}}(s_t) - G_t)^2, (V_\theta(s_t) - G_t)^2\right)\right], \tag{3.30}$$

$$V^{\text{clip}}(s_t) = V_{\theta,\text{old}}(s_t) + \text{clip}\left(V_\theta(s_t) - V_{\theta,\text{old}}(s_t), -\epsilon, \epsilon\right). \tag{3.31}$$

MAPPO extends these objectives to the multi-agent CTDE setting by using a centralized value model and decomposing the joint policy according to Eq. 3.12.

Value Decomposition

In the CTDE setting, QMIX (Rashid et al., 2018) proposes to combine the value networks $V^i(h_t^i)$ of individual agents into a centralized value model $V(s_t)$. It relies on weights $w^i(s_t)$ and bias $b(s_t)$ generated by a learnable network called mixer network:

$$V(s_t) \triangleq \sum_i w^i(s_t)V^i(h_t^i) + b(s_t). \tag{3.32}$$

3.2.3 Offline RL and Offline MARL

Offline Learning (Fujimoto et al., 2019; Levine et al., 2020) refers to the setting in which agents only have access to a fixed dataset of trajectories $\mathcal{D}$ and cannot collect additional interactions with the environment and other agents. The challenge is therefore to learn a policy that will generalize and perform well when evaluated in the environment. In order to do so, methods constrain the policy so that it stays close to the offline dataset and does not stray toward regions of the state-action space for which no interactions are available.

Implicit Q-Learning (IQL) and Multi-Agent Implicit Q-Learning (MAIQL)

IQL (Kostrikov et al., 2021) combines a SARSA approach with weighted imitation learning (see Subsection 3.3) to ensure that the policy stays close to the dataset distribution. In order to improve on the dataset policy, IQL uses a greedy and optimistic value learning scheme that estimates the e expectile of Q instead of the mean Q (this latter is a special case where $e = 0.5$). This estimation assumes that actions that yielded better outcomes will be more likely to be selected at evaluation. This makes sense given that, at evaluation, the agent will select actions so as to maximize its return. To ensure that the expectile estimation incorporates only the action selection distribution and is not influenced by the randomness of the environment's transitions, IQL uses a state-only value function V that marginalizes over future transitions:

$$\begin{aligned} L_V(\psi) &= \mathbb{E}_{s,a\sim\mathcal{D}}\left[\mathcal{L}_2^e(Q_{\hat{\theta}}(s,a) - V_\psi(s))\right] \\ &= \mathbb{E}_{s,a\sim\mathcal{D}}\left[\left|e - \mathbb{I}\left(Q_{\hat{\theta}}(s,a) - V_\psi(s) < 0\right)\right| \left(Q_{\hat{\theta}}(s,a) - V_\psi(s)\right)^2\right], \end{aligned} \tag{3.33}$$

$$L_Q(\theta) = \mathbb{E}_{s,a,r,s' \sim \mathcal{D}} \left[\left(r(s,a) + \gamma V_\psi(s') - Q_\theta(s,a) \right)^2 \right]. \tag{3.34}$$

Policy extraction is done with Advantage Weighted Regression (AWR) to favor actions with good outcomes while ensuring that they exist in the dataset:

$$L_\pi(\phi) = \mathbb{E}_{(s,a)\sim\mathcal{D}} \left[-\exp\left(\beta \left(Q_{\hat{\theta}}(s,a) - V_\psi(s) \right) \right) \log \pi_\phi(a|s) \right]. \tag{3.35}$$

We make IQL multi-agent (i.e. MAIQL) by leveraging the CTDE formalism and using QMIX value-decomposition (Rashid et al., 2018) for both Q and V:

$$\begin{aligned} V_\psi(s) &= \sum_i w_V^i(s) V_{\psi^i}(h^i) + b_V(s), \\ Q_\theta(s) &= \sum_i w_Q^i(s) Q_{\theta^i}(h^i) + b_Q(s). \end{aligned} \tag{3.36}$$

And the target network:

$$Q_{\hat{\theta}}(s) = \sum_i \hat{w}_Q^i(s) Q_{\hat{\theta}^i}(s^i) + \hat{b}_Q(s). \tag{3.37}$$

The joint policy is decomposed as Eq. 3.12. Injecting this into Eq. 3.35 one gets:

$$\begin{aligned} L_\pi(\phi) &= \mathbb{E}_{(s,a)\sim\mathcal{D}} \left[-B(s) \prod_i \exp\left(\beta(\hat{w}_Q^i(s) Q_{\hat{\theta}^i}(h^i, a^i) - w_V^i(s) V_{\psi^i}(h^i)) \right) \sum_j \log \pi_{\phi^j}(a^j|h^j) \right], \\ B(s) &= \exp(\hat{b}_Q(s) - b_V(s)). \end{aligned} \tag{3.38}$$

3.3 Imitation and Inverse Reinforcement Learning

Imitation Learning (IL) and Inverse Reinforcement Learning (IRL) both consider the case where the MDP's reward is unknown. Instead, the task at hand is specified by providing demonstrations, i.e., rollouts from an expert policy, π_E, that solves that task (Pomerleau, 1991). In some cases, the policy itself is provided (Ross and Bagnell, 2010). Imitation Learning focuses on retrieving an optimal policy (Pomerleau, 1991; Ho and Ermon, 2016) and while it does not aim at finding the MDP's reward function, it sometimes relies on inferred reward functions to guide the imitation policy (Abbeel and Ng, 2004; Ho and Ermon, 2016), that case is often referred to as Apprenticeship Learning. On the other hand, Inverse Reinforcement Learning focuses on retrieving the MDP's reward function, yet in order to do so, it usually has to train a policy (Ng et al., 2000; Fu et al., 2017).

3.3.1 Maximum Causal Entropy Inverse Reinforcement Learning

Maximum causal entropy IRL (Ziebart et al., 2008) proposes to fit a reward function r from a set $\mathcal{R}$ and retrieve the corresponding optimal policy by solving the optimization problem:

$$\min_{r\in\mathcal{R}} \left(\max_{\pi} J_{\pi}[r(s,a) + \mathcal{H}(\pi(\cdot|s))] - J_{\pi_{E}}[r(s,a)] \right). \tag{3.39}$$

In brief, the problem reduces to finding a reward function r for which the expert policy is optimal. In order to do so, the optimization procedure searches high entropy policies that are optimal with respect to r and minimizes the difference between their returns and the return of the expert policy, eventually reaching a policy π that approaches π_{E}.

Most of the proposed solutions (Abbeel and Ng (2004); Ziebart (2010) and Ho and Ermon (2016)) transpose IRL to the problem of distribution matching; Abbeel and Ng (2004) and Ziebart et al. (2008) used linear function approximation and proposed to match the feature expectation; Ho and Ermon (2016) proposed to cast Eq. (3.39) with a convex reward function regularizer into the problem of minimizing the Jensen-Shannon divergence between the state-action occupancy measures:

$$\min_{\pi} D_{\mathrm{JS}}(d_{\pi}, d_{\pi_{E}}) - J_{\pi}[\mathcal{H}(\pi(\cdot|s))]. \tag{3.40}$$

Connections between Generative Adversarial Network (GAN) and IRL.

For the data distribution p_{E} and the generator distribution p_{G} defined on the domain $\mathcal{X}$, the GAN objective (Goodfellow et al., 2014) is

$$\min_{p_{G}} \max_{D} L(D, p_{G}), \quad L(D, p_{G}) \triangleq \mathbb{E}_{x\sim p_{E}}[\log D(x)] + \mathbb{E}_{x\sim p_{G}}[\log(1 - D(x))]. \tag{3.41}$$

In Goodfellow et al. (2014), the maximizer of the inner problem in Eq. (3.41) is shown to be

$$D^{*}_{p_{G}} \triangleq \arg\max_{D} L(D, p_{G}) = \frac{p_{E}}{p_{E} + p_{G}}, \tag{3.42}$$

and the optimizer for Eq. (3.41) is

$$\arg\min_{p_{G}} \max_{D} L(D, p_{G}) = \arg\min_{p_{G}} L(D^{*}_{p_{G}}, p_{G}) = p_{E}. \tag{3.43}$$

Later, Finn et al. (2016a) and Ho and Ermon (2016) concurrently proposed connections between GANs and IRL. The Generative Adversarial Imitation Learning (GAIL) formulation in Ho and Ermon (2016) is based on matching state-action occupancy measures, while Finn et al. (2016a) considered matching trajectory distributions. Finn et al. (2016a) also proposed

the following discriminator structure:

$$D_\theta(\tau) \triangleq \frac{p_\theta(\tau)}{p_\theta(\tau) + q(\tau)}, \tag{3.44}$$

where $p_\theta(\tau) \propto \exp G_\theta(\tau)$ with reward approximator G_θ is motivated by maximum causal entropy IRL. Note that Eq. (3.44) matches the form of the optimal discriminator in Eq. (3.42). Although Finn et al. (2016a) do not empirically support the effectiveness of their method, the Adversarial Inverse Reinforcement Learning (AIRL) approach of Fu et al. (2017) successfully used a similar discriminator for state-action occupancy measure matching.

3.3.2 Offline methods for IL and IRL

Most IL and IRL methods that rely on distribution matching between learned policy statistics and demonstrated policy statistics require to frequently evaluate such statistics during training (Ng et al., 2000; Abbeel and Ng, 2004; Ziebart et al., 2008; Ho and Ermon, 2016; Fu et al., 2017). In order to do so, they need to either generate trajectories by having the policy interact with the environment or have access to the environment model: they are online in the sense that learning and environment interactions are entangled. Offline methods on the other hand only rely on the provided demonstration and do not require additional interactions.

Behavioral Cloning (BC)

BC (Pomerleau, 1991) is the simplest yet very efficient approach to IL that trains a parametrized policy $\pi(a|s;\theta)$ to maximize the log-likelihood of the data-set of demonstrated trajectories $\mathcal{D} = \{\tau_i\}_{i=0}^N$. Making the common supervised learning assumption that elements of the data-set (in this case trajectories) are independent, the negative log-likelihood of the data under policy parameters θ is given by:

$$\begin{aligned} L(\mathcal{D};\theta) &= -\log \prod_{i=0}^{N} p_\pi(\tau^i;\theta) \\ &= -\log \prod_{i=0}^{N} P_0(s_0^i) \prod_{t=0}^{T-1} \pi(a_t^i|s_t^i;\theta) P(s_{t+1}^i|s_t^i,a_t^i) \\ &= -\sum_{i=0}^{N} \log P_0(s_0^i) - \sum_{i=0}^{N}\sum_{t=0}^{T-1} \log \pi(a_t^i|s_t^i;\theta) - \sum_{i=0}^{N}\sum_{t=0}^{T-1} \log P(s_{t+1}^i|s_t^i,a_t^i). \end{aligned} \tag{3.45}$$

And the optimal parameters θ^* are given by

$$\theta^* = \arg\min_\theta L(\mathcal{D};\theta) = -\arg\min_\theta \sum_{i=0}^{N}\sum_{t=0}^{T-1} \log \pi(a_t^i|s_t^i;\theta), \tag{3.46}$$

which can be optimized through gradient descent and corresponds to either a classification or regression task depending on whether actions are discrete or continuous.

3.4 Planning

We refer here to the process by which one leverages knowledge about the world in his head to figure out the best course of action. More specifically, planning is one approach to Reinforcement Learning where the agent exploits MDP knowledge – such as transition or reward functions – so as to derive the sequence of actions that would maximize its expected return. In that regard, the Value Iteration algorithm described in Section 3.2.1 is a planning algorithm.

3.4.1 Monte Carlo Tree Search (MCTS)

MCTS addresses planning by searching over possible future trajectories to estimate the expected return of a sequence of actions (Browne et al., 2012). In order to do so, it uses MDP knowledge such as the action space $\mathcal{A}$ and the transition distribution P to sample trajectories, as well as the reward function r to compute returns.

MCTS carries out the trajectory search by constructing a search tree composed of nodes (see Figure 3.2). Following the nomenclature of Lecarpentier (2020), these nodes are either *decision nodes* ν_s – labeled with a state value – or *chance nodes* ν_s^a – labeled with a state-action pair value. The children nodes of a decision node are chances-node labeled with the same state value as their parent decision node, indeed they correspond to choosing an action from that state. Reciprocally, the children of a chance node are decision nodes labeled with the state value sampled from the transition distribution with the state-action pair of the parent chance-node, indeed they correspond to sampling a successor state from that state-action pair. Decision nodes without children are called *leaf nodes*. The *root node* of the tree is a decision node labeled with the current state of the agent. In terms of notation, x_t^i denotes the i^{th} realization of the random variable X_t. It is possible that different realizations have the same value, i.e., $x_t^i = x_t^j$, yet they only correspond to the same node if they correspond to the same trajectory, i.e., have the same parent node. When environment transitions are deterministic, chance nodes only have one child and the search tree can be built without the use of chance nodes by labeling the edges between decision nodes with the corresponding action.

One MCTS iteration corresponds to four steps: *node selection* and *node expansion* (both rely on the *tree policy*), a *simulation* phase (using a *default policy*) and the *back-propagation* step. As indicated in Figure 3.3, the tree policy moves through the tree by selecting the most promising action at each decision node it visits. If it selects an action that has never been tried before from that decision node (or if a new state is sampled from an existing chance node), it expands the tree with a new chance node (respectively decision node). Once the new leaf node is created, its value is estimated using the default policy and then this value is back-propagated through the visited nodes to update their own value estimate.

There exists several tree policies but the most common is Upper Confidence bound applied to Trees (UCT) (Kocsis and Szepesvári, 2006) that relies on Upper Confidence Bound (UCB) (Auer et al., 2002):

$$\begin{aligned} a^* &= \arg\max_a \left(Q(s,a) + 2C_p \sqrt{\frac{\log \sum_b N(s,b)}{N(s,a)}} \right), \\ Q(s,a) &= \frac{\sum_{i=0}^{N(s,a)-1} G_i(s,a)}{N(s,a)}. \end{aligned} \tag{3.47}$$

Where $N(s,a)$ counts the number of times that the chance node ν_s^a has been visited, $G_i(s,a)$ are all the returns ν_s^a has collected to this point and C_p is a constant that trades-off exploitation and exploration during the node selection and is usually set to $\sqrt{2}$.

Usually, the default policy is either a problem dependent heuristic or a Monte Carlo rollout performed from the leaf node by selecting actions at random for $H+1$ times. The corresponding Monte-Carlo Return is then:

$$v^l = \sum_{i=0}^{H} \gamma^i r_{t+l+i}, \tag{3.48}$$

where t indicates the time index of the root node and l is the depth of the leaf node inside the tree.

Finally, during the back-propagation step, all the chance nodes that were visited during this iteration collect a new return value. For a visited chance node at depth k and a expanded leaf node at depth l:

$$G^k = \sum_{\tau=0}^{l-1-k} \gamma^\tau r_{k+1+\tau} + \gamma^{l-k} v^l. \tag{3.49}$$

The budget B defines the number of MCTS iteration that are carried out before acting in the environment with action $\arg\max_a Q(s,a)$. Kocsis and Szepesvári (2006) showed that using MCTS with UCT and Monte Carlo returns converges to the optimal policy as the budget tends to infinity.

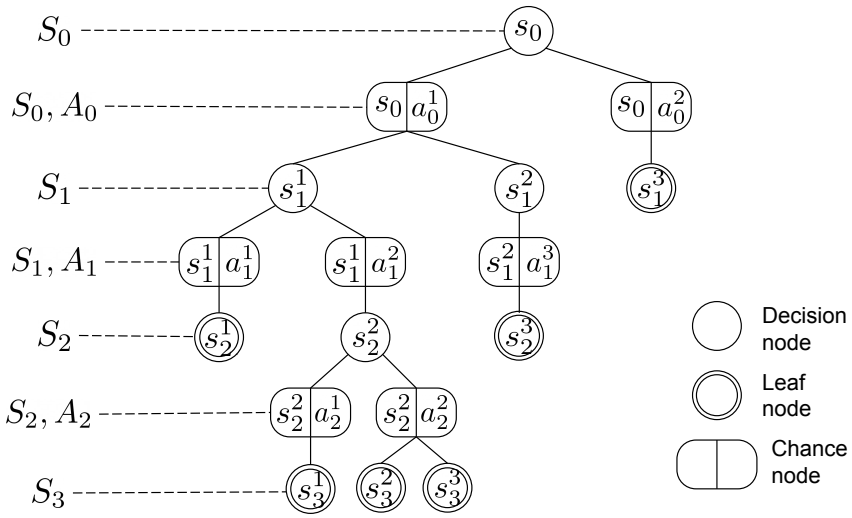


Figure 3.2: Search tree composed of decision nodes and chance nodes. The root node corresponds to the current state, in this case the initial state.

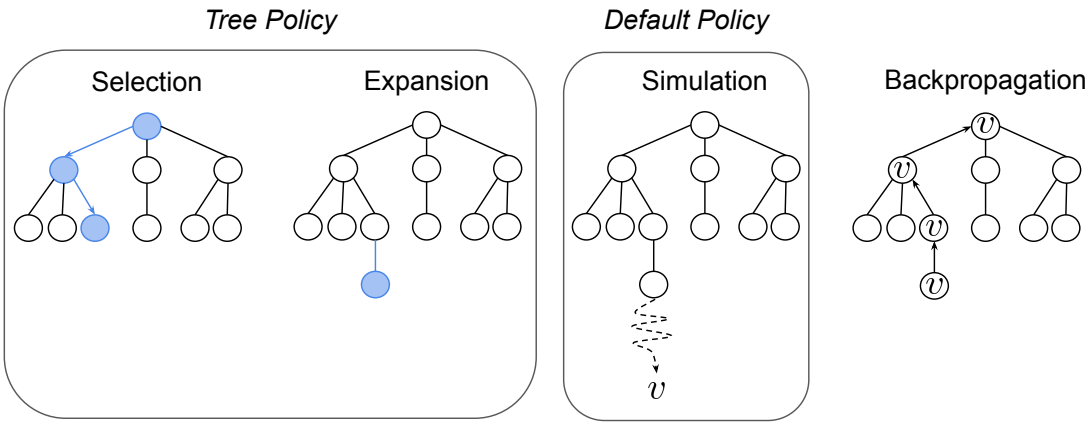


Figure 3.3: The four steps of one MCTS iteration. Illustrated in the specific case of deterministic transitions (edges are chance nodes).

Chapter 4

Improving coordination with shared incentives in Deep MARL

This chapter investigates coordination in Multi-Agent Reinforcement Learning by featuring work that we have published (Roy et al., 2020). We start by noting that prevalent MARL methods often fail to learn efficient cooperative group strategies. This is worrying and unexpected given that the MARL framework provides all the agents with rewards as well as the possibility to fine-tune to each other from interactions. One would expect these conditions to warrant coordination, yet, even when provided with perfect knowledge about their teammates during training, MARL agents tend to "ignore" one another. To remediate this, we propose to foster fruitful group strategies by relying on coordination-promoting shared incentives. First, we suggest a definition for coordination and motivate it by showcasing the efficiency of coordinated group strategies. Then, we define coordination-promoting social norms and propose to enforce these by relying on auxiliary tasks to shape the agents' interactions during learning. The first method derives directly from our definition of coordination and enforces that coordinated agents are both *predictable* and *able to predict* their teammates' behaviors. The second one leverages a centralized entity during training so that agents collectively recognize relevant situations and switch to their corresponding fine-tuned sub-policy, i.e., *synchronously* alternating between different *group strategies* depending on the situation at hand. It appears that leveraging such social norms to promote coordination greatly improves the performance of cooperative multi-agent learning.

In MARL, discovering successful collective behaviors is challenging as it requires exploring a joint action space that grows exponentially with the number of agents. While the tractability of independent agent-wise exploration is appealing, this approach fails on tasks that require elaborate group strategies. A popular framework for MARL is the use of the Centralized Training and Decentralized Execution (CTDE) procedure (Lowe et al., 2017; Foerster et al., 2018b; Iqbal and Sha, 2019; Foerster et al., 2019; Rashid et al., 2018). Typ-

ically, one leverages centralized critics to approximate the value function of the aggregated observations-action pairs and train actors restricted to the observation of a single agent. Such critics, if exposed to coordinated joint actions leading to high returns, can steer the agents' policies toward these highly rewarding behaviors. However, these approaches depend on the agents luckily stumbling on these collective actions in order to grasp their benefit. Thus, it might fail in scenarios where such behaviors are unlikely to occur by chance. We hypothesize that in such scenarios, coordination-promoting inductive biases on the policy search could help discover successful behaviors more efficiently and supersede task-specific reward shaping and curriculum learning. To motivate this proposition we present a simple Markov Game in which agents with coordinated actions learn remarkably faster. For more realistic tasks in which coordinated strategies cannot be easily engineered and must be learned, we propose to transpose this insight by relying on two coordination proxies to bias the policy search. The first avenue, TeamReg, assumes that an agent must be able to predict the behavior of its teammates in order to coordinate with them. The second, CoachReg, supposes that coordinated agents collectively recognize different situations and synchronously switch to different sub-policies to react to them.

This work makes three distinct contributions. First, we show that coordination can crucially accelerate CMAL. Second, we propose two novel approaches that aim at promoting such coordination by augmenting CTDE algorithms with additional shared multi-agent objectives (or social norms) that act as policy regularizers when optimized jointly with the main return-maximization objective. Third, we design two new sparse-reward cooperative tasks in the multi-agent particle environment (Mordatch and Abbeel, 2018). We use them along with two standard multi-agent tasks to present a detailed evaluation of our approaches' benefits when they extend the reference CTDE algorithm MADDPG (Lowe et al., 2017). We validate our methods' key components by performing ablation studies and detailed analyses of their effects on agents' behaviors. Finally, we validate that these learning advantages carry on to the Google Research Football environment (Kurach et al., 2019), which involves discrete actions and greater complexity.

Based on our experiments and analysis, the TeamReg objective seems to provide a dense learning signal that can help guide the policy towards coordination in the absence of external reward, eventually leading to the discovery of higher-performing team strategies in a number of cooperative tasks. However, we also find that TeamReg does not lead to improvements in every single case and can even be harmful in environments with a competitive component. CoachReg enforces synchronous sub-policy selection which enables the agents to concurrently learn to react to different agreed-upon situations. This consistently improves the group's performance.

4.1 Motivation: coordinated agents learn better

Can coordination help the discovery of effective policies in cooperative tasks? Intuitively, coordination can be defined as an agent's behavior being informed by the behavior of another agent, i.e. structure in the agents' interactions. Namely, a team where agents behave independently of one another would not be coordinated.

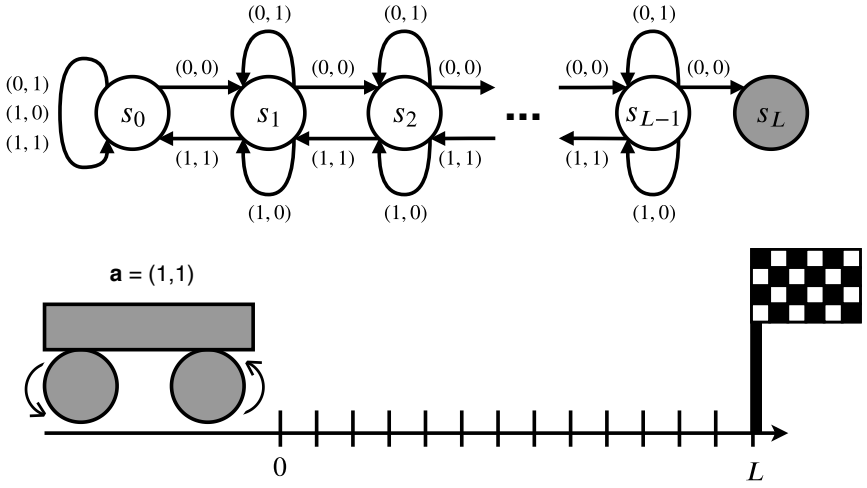


Figure 4.1: Simple Markov Game consisting of a chain of length L leading to a terminal state (in grey). Agents can be seen as the two wheels of a vehicle so their actions need to be in agreement for the vehicle to move.

Consider the simple Markov Game consisting of a chain of length L leading to a termination state as depicted in Figure 4.1. At each time step, both agents receive $r_t = -1$. The joint action of these two agents in this environment is given by $\mathbf{a} \in \mathcal{A} = \mathcal{A}^1 \times \mathcal{A}^2$, where $\mathcal{A}^1 = \mathcal{A}^2 = \{0, 1\}$. Agent i tries to go right when selecting $a^i = 0$ and left when selecting $a^i = 1$. However, to transition to a different state, both agents need to perform the same action at the same time (two lefts or two rights). Now consider a slight variant of this environment with a different joint action structure $\mathbf{a}' \in \mathcal{A}'$. The former action structure is augmented with a hard-coded coordination module that maps the joint primitive a^i to $a^{i\prime}$ like so:

$$\mathbf{a}' = \begin{pmatrix} a^{1\prime} = a^1 \\ a^{2\prime} = a^1 a^2 + (1 - a^1)(1 - a^2) \end{pmatrix}, \begin{pmatrix} a^1 \\ a^2 \end{pmatrix} \in \mathcal{A}$$

While the second agent still learns a state-action value function $Q^2(s, a^2)$ with $a^2 \in \mathcal{A}^2$, the coordination module builds $a^{2\prime}$ from (a^1, a^2) so that a^2 effectively determines whether the second agent acts in *agreement* or in *disagreement* with the first agent. In other words, if $a^2 = 1$, then $a^{2\prime} = a^1$ (agreement) and if $a^2 = 0$, then $a^{2\prime} = 1 - a^1$ (disagreement).

While it is true that this additional structure does not modify the action space nor the independence of the action selection, it reduces the stochasticity of the transition dynamics as seen by agent 2. In the first setup, the outcome of an agent's action is conditioned on the action of the other agent. In the second setup, if agent 2 decides to disagree, the transition becomes deterministic as the outcome is independent of agent 1. This suggests that by reducing the entropy of the transition distribution, this mapping reduces the variance of the Q-updates and thus makes online tabular Q-learning agents learn much faster (Figure 4.2).

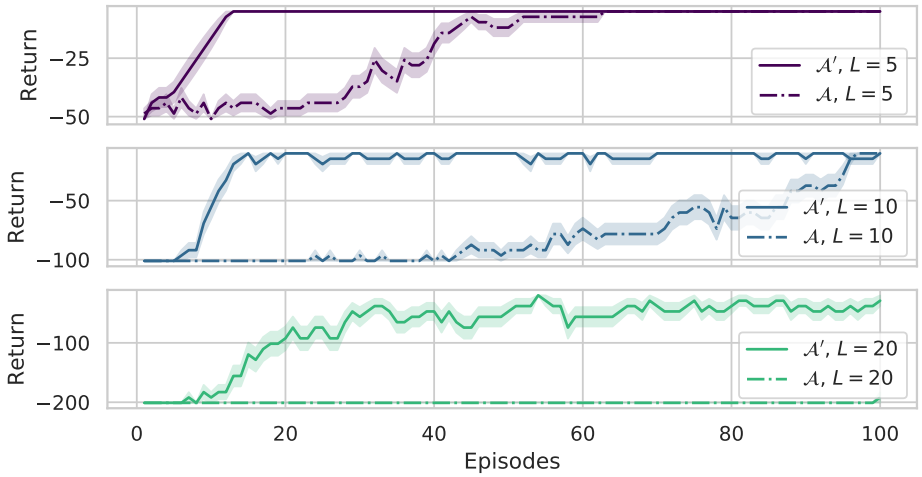


Figure 4.2: Learning curves for tabular Q-learning. Agents learn much more efficiently when constrained to the space of coordinated policies (solid lines) than in the original action space (dashed lines). We trained each agent i with online Q-learning (Watkins and Dayan, 1992) on the $Q^i(a^i, s)$ table using Boltzmann exploration (Kaelbling et al., 1996). The Boltzmann temperature is fixed to 1 and we set the learning rate to 0.05 and the discount factor to 0.99. After each learning episode, we evaluate the current greedy policy on 10 episodes and report the mean return. Curves are averaged over 20 seeds and the shaded area represents the standard error.

In this section, we used handcrafted mappings in order to demonstrate the effectiveness of exploring the space of coordinated policies rather than the unconstrained policy space. The following sections present two approaches to softly learning the same type of constraints throughout training for any multi-agent cooperative tasks.

4.2 Promoting coordination

Here, we investigate techniques to enforce inductive biases that foster coordination. We propose to rely on shared social norms and implement these using auxiliary tasks to regularize the agents' policies. We devise two methods: TeamReg, which is based on inter-agent action predictability, and CoachReg which relies on synchronized behavior selection.

4.2.1 Team regularization

We propose to exploit the structure present in the joint action space of coordinated policies to attain a certain degree of predictability of one agent's behavior with respect to its teammates. It is based on the hypothesis that the reciprocal also holds i.e., that promoting agents' predictability could foster such team structure and lead to more coordinated behaviors. This assumption is cast into the decentralized framework by training agents to predict their teammates' actions given only their own observations. For continuous control, the loss is the Mean Squared Error (MSE) between the predicted and true actions of the teammates, yielding a teammate-modeling secondary objective. For discrete action spaces, we use the Kullback–Leibler (KL) divergence (D_{KL}) between the predicted and real action distributions.

While estimating teammates' policies can be used to enrich the learned representations, we extend this objective to also drive the teammates' behaviors towards the predictions by leveraging a differentiable action selection mechanism. This means that gradients are backpropagated through both the prediction and the real action selection modules. We call *team-spirit* the objective pair $J^{i,j}_{TS}$ and $J^{j,i}_{TS}$ between agents i and j:

$$J^{i,j}_{TS\text{-continuous}}(\theta^i, \theta^j) = -\mathbb{E}_{\mathbf{o}_t \sim \mathcal{D}} \left[\text{MSE}(\mu^j(o^j_t; \theta^j), \hat{\mu}^{i,j}(o^i_t; \theta^i)) \right] \tag{4.1}$$

$$J^{i,j}_{TS\text{-discrete}}(\theta^i, \theta^j) = -\mathbb{E}_{\mathbf{o}_t \sim \mathcal{D}} \left[\text{D}_{\text{KL}} \left(\pi^j(\cdot|o^j_t; \theta^j) || \hat{\pi}^{i,j}(\cdot|o^i_t; \theta^i) \right) \right] \tag{4.2}$$

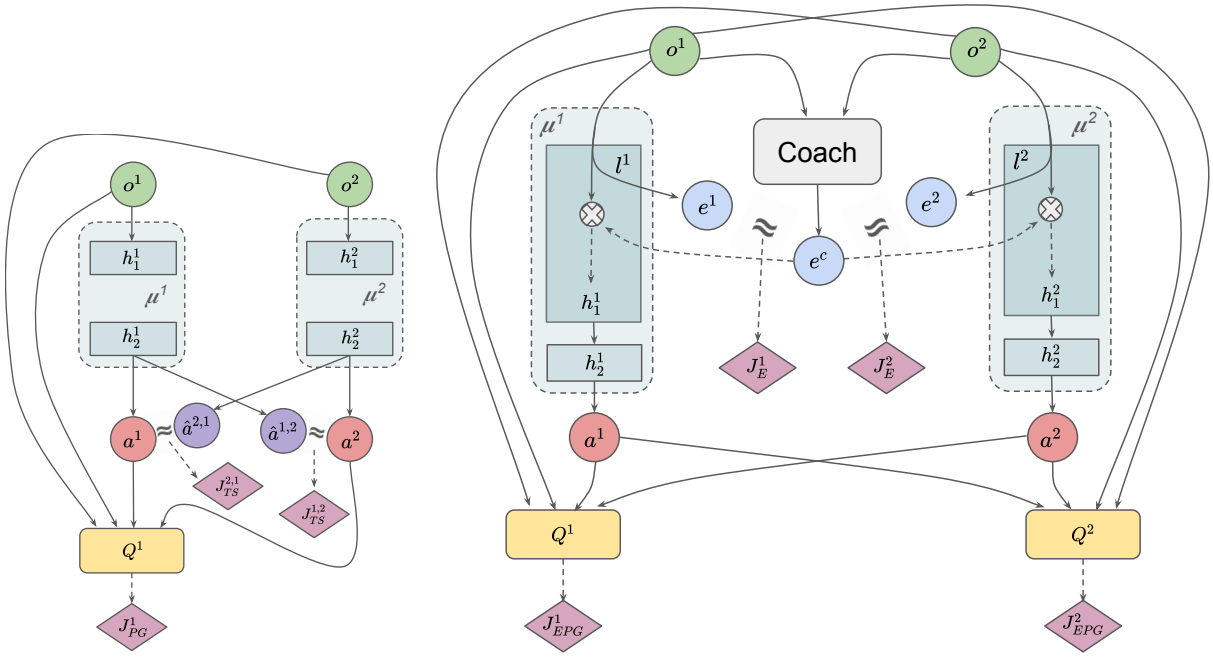
where $\hat{\mu}^{i,j}$ (or $\hat{\pi}^{i,j}$ in the discrete case) is the policy head of agent i trying to predict the action of agent j. The total objective for a given agent i becomes:

$$J^i_{total}(\theta^i) = J^i_{PG}(\theta^i) + \lambda_1 \sum_j J^{i,j}_{TS}(\theta^i, \theta^j) + \lambda_2 \sum_j J^{j,i}_{TS}(\theta^j, \theta^i) \tag{4.3}$$

where λ_1 and λ_2 are hyperparameters that respectively weigh how well an agent should predict its teammates' actions, and how predictable an agent should be for its teammates. We call TeamReg this dual regularization from team-spirit objectives. Figure 4.3 summarizes the method.

4.2.2 Coach regularization

This method aims at teaching agents to recognize different situations and synchronously select corresponding sub-behaviors to foster coordinated interactions.



(a) TeamReg (b) CoachReg

Figure 4.3: Illustration of the policy regularization methods with two agents. (a) TeamReg: each agent's policy is equipped with additional heads that are trained to predict other agents' actions and every agent is regularized to produce actions that its teammates correctly predict. The method is depicted for agent 1 only to avoid cluttering. (b) CoachReg: a central model, the coach, takes all agents' observations as input and outputs the current mode (policy mask). Agents are regularized to predict the same mask from their local observations and optimize the corresponding sub-policy.

Sub-policy selection

Explicit sub-behavior selection is implemented by relying on *policy masks* to modulate the agent's policy. A policy mask u^j is a one-hot vector of size K (a fixed hyperparameter) with its j^{th} component set to one. In practice, we use these masks to perform dropout (Srivastava et al., 2014) in a structured manner on $\tilde{h}_1 \in \mathbb{R}^M$, the pre-activations of the first hidden layer h_1 of the policy network π. To do so, we construct the vector $\boldsymbol{u}^j$, which is the concatenation of C copies of u^j, in order to reach the dimensionality $M = C * K$. The element-wise product $\boldsymbol{u}^j \odot \tilde{h}_1$ is performed and only the units of $\tilde{h}_1$ at indices m modulo $K = j$ are kept for $m = 0, \ldots, M-1$. Each agent i generates e_t^i, its own policy mask from its observation o_t^i,

to modulate its policy network. Here, a simple linear layer l^i is used to produce a categorical probability distribution $p^i(e_t^i|o_t^i;\theta^i)$ from which the one-hot vector is sampled:

$$p^i(e_t^i = u^j|o_t^i;\theta^i) = \frac{\exp\left(l^i(o_t^i;\theta^i)_j\right)}{\sum_{k=0}^{K-1}\exp\left(l^i(o_t^i;\theta^i)_k\right)} \tag{4.4}$$

Synchronous and coherent sub-policy selection

Agents can swiftly switch between sub-policies by using policy masking, however, they are not encouraged to modulate their behavior synchronously and coherently. To promote this, we introduce the *coach* entity, parametrized by ψ, which learns to produce policy masks e_t^c from the joint observations, i.e. $p^c(e_t^c|\mathbf{o}_t;\psi)$. The coach is used at training time only and drives the agents toward synchronously selecting the same behavior mask. Specifically, the coach is trained to output masks that (1) yield high returns when used by the agents, and (2), are predictable by the agents. Similarly, each agent is regularized so that (1) its private mask matches the coach's mask, and (2), it derives efficient behavior when using the coach's mask. At evaluation time, the coach is removed and the agents only rely on their own policy masks. The policy gradient objective when agent i is provided with the coach's mask is given by:

$$J^i_{EPG}(\theta^i,\psi) = \mathbb{E}_{\mathbf{o}_t,\mathbf{a}_t\sim\mathcal{D}}\left[Q^i(\mathbf{o}_t,\mathbf{a}_t)\Big|_{\substack{a_t^i=\mu(o_t^i,e_t^c;\theta^i)\\ e_t^c\sim p^c(\cdot|\mathbf{o}_t;\psi)}}\right] \tag{4.5}$$

The difference between the mask distribution of agent i and the coach's one is measured from the KL divergence:

$$J^i_E(\theta^i,\psi) = -\mathbb{E}_{\mathbf{o}_t\sim\mathcal{D}}\left[\mathrm{D}_{\mathrm{KL}}\left(p^c(\cdot|\mathbf{o}_t;\psi)|\,|p^i(\cdot|o_t^i;\theta^i)\right)\right] \tag{4.6}$$

The total objective for agent i is:

$$J^i_{total}(\theta^i) = J^i_{PG}(\theta^i) + \lambda_1 J^i_E(\theta^i,\psi) + \lambda_2 J^i_{EPG}(\theta^i,\psi) \tag{4.7}$$

with λ_1 and λ_2 the regularization coefficients. Similarly, the coach is trained with the following dual objective, weighted by the λ_3 coefficient:

$$J^c_{total}(\psi) = \frac{1}{N}\sum_{i=1}^{N}\left(J^i_{EPG}(\theta^i,\psi) + \lambda_3 J^i_E(\theta^i,\psi)\right) \tag{4.8}$$

In order to propagate gradients through the sampled policy mask we reparameterize the categorical distribution using the Gumbel-softmax (Jang et al., 2017) with a temperature of 1. We call this coordinated sub-policy selection regularization CoachReg and illustrate it in Figure 4.3.

4.3 Experimental Setup

This section presents our experimental design. We describe the continuous control tasks with sparse rewards that we used to test our approach. We also detail the thorough hyperparameter search and evaluation process that the followed to ensure fair and informative comparisons with the baselines.

4.3.1 Training environments

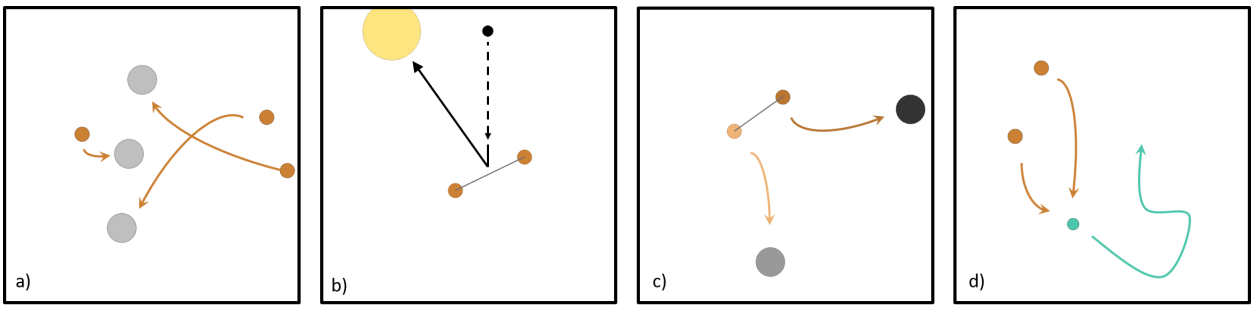


Figure 4.4: The multi-agent tasks we employ. (a) SPREAD: Agents must spread out and cover a set of landmarks. (b) BOUNCE: Two agents are linked together by a spring and must position themselves so that the falling black ball bounces towards a target. (c) COMPROMISE: Two linked agents must compete or cooperate to reach their own assigned landmark. (d) CHASE: Two agents chase a (non-learning) prey (turquoise) that moves w.r.t repulsion forces from predators and walls.

We build novel continuous control tasks in OpenAI's multi-agent particle environment (Mordatch and Abbeel, 2018). SPREAD and CHASE were introduced by Lowe et al. (2017). We use SPREAD as is but with sparse rewards. CHASE is modified with a prey controlled by repulsion forces so that only the predators are learnable, as we wish to focus on coordination in cooperative tasks. Finally, we introduce COMPROMISE and BOUNCE where agents are physically tied together. While positive return can be achieved in these tasks by selfish agents, they all benefit from coordinated strategies and maximal return can only be achieved by agents working closely together. Figure 4.4 presents a visualization and a brief description. In all tasks, agents receive as observation their own global position and velocity as well as the relative position of other entities. Note that works showcasing experiments in these environments often use discrete action spaces and dense rewards (e.g. the proximity with the objective) (Iqbal and Sha, 2019; Lowe et al., 2017; Jiang and Lu, 2018). In our experiments, agents learn with continuous action spaces and from sparse rewards which is a far more challenging setting.

Tasks description

SPREAD (Figure 4.4a): In this environment, there are 3 agents (small orange circles) and 3 landmarks (bigger gray circles). At every timestep, agents receive a team-reward $r_t = n - c$ where n is the number of landmarks occupied by at least one agent and c is the number of collisions occurring at that timestep. To maximize their return, agents must therefore spread out and cover all landmarks. Initial agents' and landmarks' positions are random. Termination is triggered when the maximum number of timesteps is reached.

BOUNCE (Figure 4.4b): In this environment, two agents (small orange circles) are linked together with a spring that pulls them toward each other when stretched above its relaxation length. At episode's mid-time a ball (smaller black circle) falls from the top of the environment. Agents must position correctly so as to have the ball bounce on the spring towards the target (bigger beige circle), which turns yellow if the ball's bouncing trajectory passes through it. They receive a team reward of $r_t = 0.1$ if the ball reflects towards the side walls, $r_t = 0.2$ if the ball reflects towards the top of the environment, and $r_t = 10$ if the ball reflects towards the target. At initialization, the target's and ball's vertical position is fixed, and their horizontal positions are random. Agents' initial positions are also random. Termination is triggered when the ball is bounced by the agents or when the maximum number of timesteps is reached.

COMPROMISE (Figure 4.4c): In this environment, two agents (small orange circles) are linked together with a spring that pulls them toward each other when stretched above its relaxation length. They both have a distinct assigned landmark (light gray circle for the light orange agent, dark gray circle for the dark orange agent), and receive a reward of $r_t = 10$ when they reach it. Once a landmark is reached by its corresponding agent, the landmark is randomly relocated in the environment. The initial positions of agents and landmarks are random. Termination is triggered when the maximum number of timesteps is reached.

CHASE (Figure 4.4d): In this environment, two predators (orange circles) are chasing a prey (turquoise circle). The prey moves with respect to a scripted policy consisting of repulsion forces from the walls and predators. At each timestep, the learning agents (predators) receive a team reward of $r_t = n$ where n is the number of predators touching the prey. The prey has a greater max speed and acceleration than the predators. Therefore, to maximize their return, the two agents must coordinate in order to squeeze the prey into a corner or a wall and effectively trap it there. Termination is triggered when the maximum number of time steps is reached.

The presented tasks are challenging and require different levels of coordination.

4.3.2 Algorithms and Training details

Algorithms

Algorithm 1 TeamReg

Randomly initialize N critic networks Q^i and actor networks μ^i
Initialize the target weights
Initialize one replay buffer $\mathcal{D}$
for episode from 0 to number of episodes **do**
  Initialize random processes $\mathcal{N}^i$ for action exploration
  Receive initial joint observation $\mathbf{o}_0$
  for timestep t from 0 to episode length **do**
    Select action $a_i = \mu^i(o_t^i) + \mathcal{N}_t^i$ for each agent
    Execute joint action $\mathbf{a}_t$ and observe joint reward $\mathbf{r}_t$ and new observation $\mathbf{o}_{t+1}$
    Store transition $(\mathbf{o}_t, \mathbf{a}_t, \mathbf{r}_t, \mathbf{o}_{t+1})$ in $\mathcal{D}$
  Sample a random minibatch of M transitions from $\mathcal{D}$
  for each agent i **do**
    Evaluate L^i and J_{PG}^i from Equations (3.23) and (3.24)
    for each other agent $(j \neq i)$ **do**
      Evaluate $J_{TS}^{i,j}$ from Equations (4.1, 4.2)
      Update actor j with $\theta^j \leftarrow \theta^j + \alpha_\theta \nabla_{\theta^j} \lambda_2 J_{TS}^{i,j}$
    Update critic with $\phi^i \leftarrow \phi^i - \alpha_\phi \nabla_{\phi^i} L^i$
    Update actor i with $\theta^i \leftarrow \theta^i + \alpha_\theta \nabla_{\theta^i} \left(J_{PG}^i + \lambda_1 \sum_{j=1}^N J_{TS}^{i,j} \right)$
  Update all target weights

Algorithm 2 CoachReg

Randomly initialize N critic networks Q^i, actor networks μ^i and one coach network p^c
Initialize N target networks $Q^{i\prime}$ and $\mu^{i\prime}$
Initialize one replay buffer $\mathcal{D}$
for episode from 0 to number of episodes **do**
  Initialize random processes $\mathcal{N}^i$ for action exploration
  Receive initial joint observation $\mathbf{o}_0$
  for timestep t from 0 to episode length **do**
    Select action $a_i = \mu^i(o_t^i) + \mathcal{N}_t^i$ for each agent
    Execute joint action $\mathbf{a}_t$ and observe joint reward $\mathbf{r}_t$ and new observation $\mathbf{o}_{t+1}$
    Store transition $(\mathbf{o}_t, \mathbf{a}_t, \mathbf{r}_t, \mathbf{o}_{t+1})$ in $\mathcal{D}$
  Sample a random minibatch of M transitions from $\mathcal{D}$
  for each agent i **do**
    Evaluate $\mathcal{L}^i$ and J_{PG}^i from Equations (3.23) and (3.24)
    Update critic with $\phi^i \leftarrow \phi^i - \alpha_\phi \nabla_{\phi^i} \mathcal{L}^i$
    Update actor with $\theta^i \leftarrow \theta^i + \alpha_\theta \nabla_{\theta^i} J_{PG}^i$
  for each agent i **do**
    Evaluate J_E^i and J_{EPG}^i from Equations (4.6) and (4.5)
    Update actor with $\theta^i \leftarrow \theta^i + \alpha_\theta \nabla_{\theta^i} \left(\lambda_1 J_E^i + \lambda_2 J_{EPG}^i\right)$
  Update coach with $\psi \leftarrow \psi + \alpha_\psi \nabla_\psi \frac{1}{N} \sum_{i=1}^N \left(J_{EPG}^i + \lambda_3 J_E^i\right)$
  Update all target weights

Generalities

In all of our experiments, we use the Adam optimizer (Kingma and Ba, 2014) to perform parameter updates. All models (actors, critics, and the coach) are parametrized by feedforward networks containing two hidden layers of 128 units. We use the Rectified Linear Unit (ReLU) (Nair and Hinton, 2010) activation function and layer normalization (Ba et al., 2016) on the pre-activations unit to stabilize the learning. We use a buffer size of 10^6 entries and a batch size of 1024. We collect 100 transitions by interacting with the environment for each learning update. For all tasks in our hyperparameter searches, we train the agents for $15,000$ episodes of 100 steps and then re-train the best configuration for each algorithm-environment pair for twice as long ($30,000$ episodes) to ensure full convergence for the final evaluation. The scale of the exploration noise is kept constant for the first half of the training time and then decreases linearly to 0 until the end of training. We use a discount factor γ of 0.95 and a gradient clipping threshold of 0.5 in all experiments. Finally, for CoachReg, we fixed K to 4, meaning that agents could choose between 4 sub-policies. Since policies' hidden layers are of size 128 the corresponding value for C is 32. All experiments were run on Intel E5-2683 v4 Broadwell (2.1GHz) CPUs in less than 12 hours.

Hyperparameter search

Here we present the thorough parameter search with which we tuned the baselines and our methods alike. Using the same budget and evaluation protocol ensures a fair and informative comparison.

Hyperparameter search ranges. We perform searches over the following hyperparameters: the learning rate of the actor α_θ, the learning rate of the critic ω_ϕ relative to the actor ($\alpha_\phi = \omega_\phi * \alpha_\theta$), the target-network soft-update parameter τ and the initial scale of the exploration noise η_{noise} for the Ornstein-Uhlenbeck noise generating process (Uhlenbeck and Ornstein, 1930) as used by Lillicrap et al. (2015). When using TeamReg and CoachReg, we additionally search over the regularization weights λ_1, λ_2, and λ_3. In order to reduce the search space, the learning rate of the coach is always equal to the actor's learning rate (i.e. $\alpha_\theta = \alpha_\psi$), motivated by their similar architectures and learning signals. Table 4.1 shows the ranges from which values for the hyperparameters are drawn uniformly during the searches.

HYPERPARAMETER	RANGE
$\log(\alpha_\theta)$	$[-8, -3]$
$\log(\omega_\phi)$	$[-2, \quad 2]$
$\log(\tau)$	$[-3, -1]$
$\log(\lambda_1)$	$[-3\,, \quad 0]$
$\log(\lambda_2)$	$[-3\,, \quad 0]$
$\log(\lambda_3)$	$[-1\,, \quad 1]$
η_{noise}	$[0.3, 1.8]$

Table 4.1: Ranges for hyperparameter search, the log base is 10

Model selection. During training, a policy is evaluated on a set of 10 different episodes every 100 learning steps. At the end of the training, the model at the best evaluation iteration is saved as the best version of the policy for this training and is re-evaluated on 100 different episodes to have a better assessment of its final performance. The performance of a hyperparameter configuration is defined as the average performance (across seeds) of the best policies learned using this set of hyperparameter values.

Hyperparameter search results. The performance distributions across hyperparameters configurations for each algorithm on each task are depicted in Figure 4.5 using a box-and-whisker plot. It can be seen that, while most algorithms can perform reasonably well with the correct configuration, TeamReg, CoachReg, as well as their ablated versions, boost the performance

of the third quartile, suggesting an increase in the robustness across hyperparameter compared to the baselines.

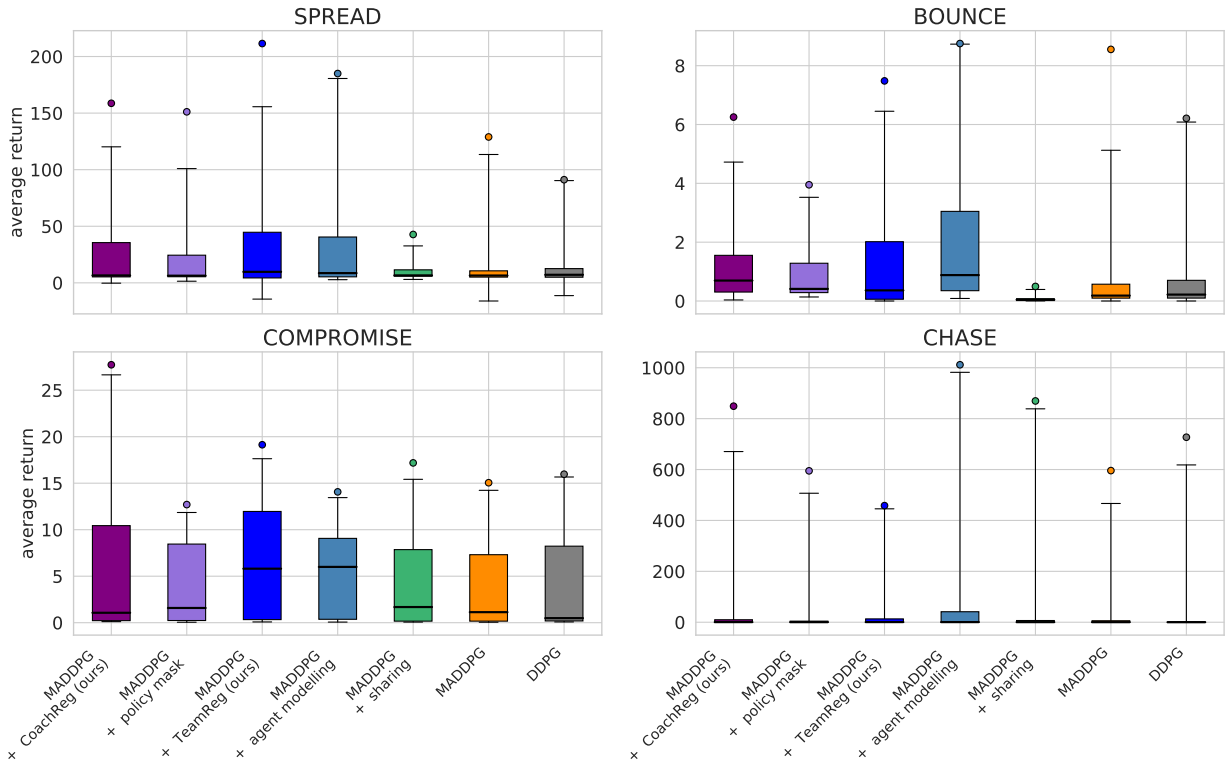


Figure 4.5: Hyperparameter tuning results for all algorithms. There is one distribution per *(algorithm, environment)* pair, each one formed of 50 data points (hyperparameter configuration samples). Each point represents the best model performance averaged over 100 evaluation episodes and averaged over the 3 training seeds for one sampled hyperparameters configuration. The box plots divide into quartiles the 49 lower-performing configurations for each distribution while the score of the best-performing configuration is highlighted above the box plots by a single dot.

Selected hyperparameters. Tables 4.2, 4.3, 4.4, and 4.5 show the best hyperparameters found by the random searches for each of the environments and each of the algorithms.

Hyperparameter	DDPG	MADDPG	MADDPG+Sharing	MADDPG+TeamReg	MADDPG+CoachReg
α_θ	$5.3 * 10^{-5}$	$2.1 * 10^{-5}$	$9.0 * 10^{-4}$	$2.5 * 10^{-5}$	$1.2 * 10^{-5}$
ω_ϕ	53	79	0.71	42	82
τ	0.05	0.083	0.076	0.098	0.0077
λ_1	-	-	-	0.054	0.13
λ_2	-	-	-	0.29	0.24
λ_3	-	-	-	-	8.4
η_{noise}	1.0	0.5	0.7	1.2	1.6

Table 4.2: Best found hyperparameters for the SPREAD environment

Hyperparameter	DDPG	MADDPG	MADDPG+Sharing	MADDPG+TeamReg	MADDPG+CoachReg
α_θ	$8.1 * 10^{-4}$	$3.8 * 10^{-5}$	$1.2 * 10^{-4}$	$1.3 * 10^{-5}$	$6.8 * 10^{-5}$
ω_ϕ	2.4	87	0.47	85	9.4
τ	0.089	0.016	0.06	0.055	0.02
λ_1	-	-	-	0.06	0.0066
λ_2	-	-	-	0.0026	0.23
λ_3	-	-	-	-	0.34
η_{noise}	1.2	0.9	1.2	1.0	1.1

Table 4.3: Best found hyperparameters for the BOUNCE environment

Hyperparameter	DDPG	MADDPG	MADDPG+Sharing	MADDPG+TeamReg	MADDPG+CoachReg
α_θ	$4.5 * 10^{-4}$	$2.0 * 10^{-4}$	$9.7 * 10^{-4}$	$1.3 * 10^{-5}$	$1.8 * 10^{-4}$
ω_ϕ	32	64	0.79	85	90
τ	0.031	0.021	0.032	0.055	0.011
λ_1	-	-	-	0.06	0.0069
λ_2	-	-	-	0.0026	0.86
λ_3	-	-	-	-	0.76
η_{noise}	0.6	1.0	1.5	1.0	1.1

Table 4.4: Best found hyperparameters for the CHASE environment

Hyperparameter	DDPG	MADDPG	MADDPG+Sharing	MADDPG+TeamReg	MADDPG+CoachReg
α_θ	$6.1 * 10^{-5}$	$3.1 * 10^{-4}$	$6.2 * 10^{-4}$	$1.5 * 10^{-5}$	$3.4 * 10^{-4}$
ω_ϕ	1.7	0.94	0.58	90	29
τ	0.065	0.045	0.007	0.02	0.0037
λ_1	-	-	-	0.0013	0.65
λ_2	-	-	-	0.56	0.5
λ_3	-	-	-	-	1.3
η_{noise}	1.1	0.7	1.3	1.6	1.6

Table 4.5: Best found hyperparameters for the COMPROMISE environment

Hyperparameter	MADDPG	MADDPG+Sharing	MADDPG+TeamReg	MADDPG+CoachReg
α_θ	$1.6 * 10^{-6}$	$3.4 * 10^{-5}$	$3.5 * 10^{-6}$	$9.4 * 10^{-5}$
ω_ϕ	3.1	13	0.96	2.9
τ	0.004	0.0014	0.0066	0.018
λ_1	-	-	0.1	0.027
λ_2	-	-	0.02	0.027
λ_3	-	-	-	2.4

Table 4.6: Best found hyperparameters for the *3-vs-1-with-keeper* Google Football environment

Selected hyperparameters (ablations). Tables 4.7, 4.8, 4.9, and 4.10 show the best hyperparameters found by the random searches for each of the environments and each of the ablated algorithms.

Hyperparameter	MADDPG+Agent Modelling	MADDPG+Policy Mask
α_θ	$1.3 * 10^{-5}$	$6.8 * 10^{-5}$
ω_ϕ	85	9.4
τ	0.055	0.02
λ_1	0.06	0
λ_2	0	0
λ_3	-	0
η_{noise}	1.0	1.1

Table 4.7: Best found hyperparameters for the SPREAD environment

Hyperparameter	MADDPG+Agent Modelling	MADDPG+Policy Mask
α_θ	$1.3 * 10^{-5}$	$2.5 * 10^{-4}$
ω_ϕ	85	0.52
τ	0.055	0.0077
λ_1	0.06	0
λ_2	0	0
λ_3	-	0
η_{noise}	1.0	1.3

Table 4.8: Best found hyperparameters for the BOUNCE environment

Hyperparameter	MADDPG+Agent Modelling	MADDPG+Policy Mask
α_θ	$2.5 * 10^{-5}$	$6.8 * 10^{-5}$
ω_ϕ	42	9.4
τ	0.098	0.02
λ_1	0.054	0
λ_2	0	0
λ_3	-	0
η_{noise}	1.2	1.1

Table 4.9: Best found hyperparameters for the CHASE environment

Hyperparameter	MADDPG+Agent Modelling	MADDPG+Policy Mask
α_θ	$1.2 * 10^{-4}$	$2.5 * 10^{-4}$
ω_ϕ	0.71	0.52
τ	0.0051	0.0077
λ_1	0.0075	0
λ_2	0	0
λ_3	-	0
η_{noise}	1.8	1.3

Table 4.10: Best found hyperparameters for the COMPROMISE environment

4.4 Results and Analysis

The approaches investigated in this work offer a way to incorporate new inductive biases in CTDE multi-agent policy search algorithms. We evaluate them by extending MADDPG, one of the most widely used algorithms in the MARL literature. We compare against vanilla MADDPG as well as two of its variants in the four cooperative multi-agent tasks described in Section 4.3.1. The first variant (DDPG) is the single-agent counterpart of MADDPG (decentralized training). The second (MADDPG + sharing) shares the policy and value-function models across agents. In addition to the two proposed algorithms and the three baselines, we present results for two ablated versions of our methods. The first ablation (MADDPG + agent modeling) is similar to TeamReg but with $\lambda_2 = 0$, which results in only enforcing agent modeling and not encouraging agent predictability. The second ablation (MADDPG + policy mask) uses the same policy architecture as CoachReg, but with $\lambda_{1,2,3} = 0$, which means that agents still predict and apply a mask to their own policy, but synchronicity is not encouraged.

To offer a fair comparison between all methods, the hyperparameter search routine is the same for each algorithm and environment (see Subsection 4.3.2). For each search-experiment pair (one per algorithm per environment), 50 randomly sampled hyperparameter configurations each using 3 random seeds are used to train the models for $15,000$ episodes. For each

algorithm-environment pair, we then select the best hyperparameter configuration for the final comparison and retrain them on 10 seeds for twice as long. This thorough evaluation procedure represents around 3 CPU-year. We give all details about the training setup and model selection in Subsection 4.3.2 and 4.3.2. The results of the hyperparameter searches are given in Subsection 4.3.2. Interestingly, Figure 4.5 shows that our proposed coordination regularizers improve robustness to hyperparameters despite having more hyperparameters to tune.

4.4.1 Asymptotic Performance

Figure 4.6 reports the average learning curves and Table 4.11 presents the final performance.

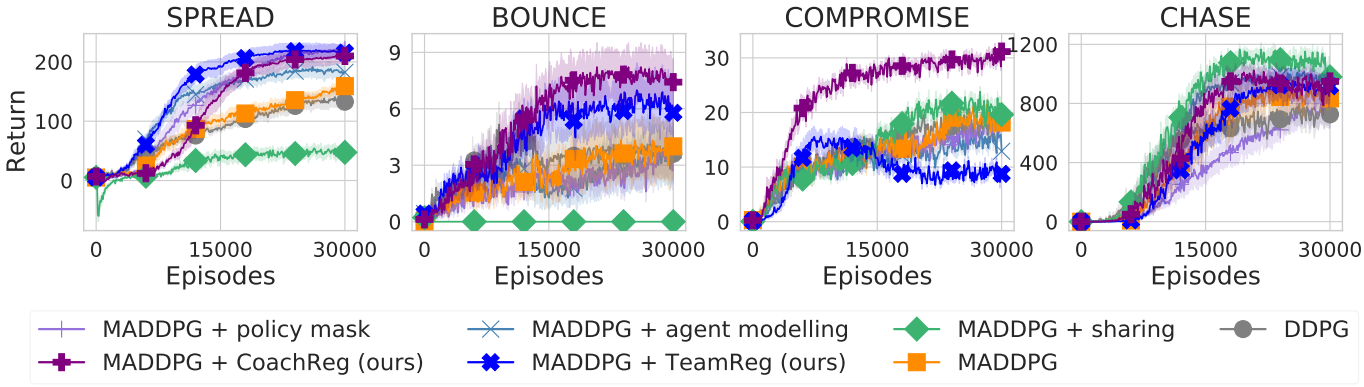


Figure 4.6: Learning curves (mean return over agents) for our two proposed algorithms, two ablations, and three baselines on all four environments. Solid lines are the mean and envelopes are the Standard Error of the Mean (SEM) across the 10 training seeds.

Algorithms \ Tasks	SPREAD	BOUNCE	COMPROMISE	CHASE
DDPG	133 ± 12	3.6 ± 1.4	19.1 ± 1.2	727 ± 87
MADDPG	159 ± 6	4.0 ± 1.6	18.1 ± 1.1	834 ± 80
+sharing	47 ± 8	0.0 ± 0.0	19.6 ± 1.5	$\mathbf{980 \pm 64}$
+agent modeling	183 ± 10	3.8 ± 1.5	12.9 ± 0.9	$\mathbf{946 \pm 69}$
+policy mask	$\mathbf{221 \pm 11}$	3.7 ± 1.1	18.4 ± 1.3	722 ± 82
+TeamReg (ours)	$\mathbf{216 \pm 12}$	$\mathbf{5.8 \pm 1.3}$	8.8 ± 0.9	$\mathbf{917 \pm 90}$
+CoachReg (ours)	$\mathbf{210 \pm 12}$	$\mathbf{7.4 \pm 1.2}$	$\mathbf{31.1 \pm 1.1}$	$\mathbf{949 \pm 54}$

Table 4.11: Final performance reported as the mean return over agents averaged across 10 episodes and 10 seeds ($\pm$ SEM).

CoachReg is the best-performing algorithm considering performance across all tasks. TeamReg also significantly improves performance on two tasks (SPREAD and BOUNCE) but shows unstable behavior on COMPROMISE, the only task with an adversarial component. This result reveals one limitation of this approach and is discussed in detail in Subsection 4.4.2. Note that all algorithms perform similarly well on CHASE, with a slight advantage to the one using parameter sharing; yet this superiority is restricted to this task where the optimal strategy is to move symmetrically and squeeze the prey into a corner. Contrary to popular belief, we find that MADDPG almost never significantly outperforms DDPG in these sparse reward environments, supporting the hypothesis that while CTDE algorithms can in principle identify and reinforce highly rewarding coordinated behavior, they are likely to fail to do so if not incentivized to coordinate.

Regarding the ablated versions of our methods, the use of unsynchronized policy masks might result in swift and unpredictable behavioral changes and make it difficult for agents to perform together and coordinate. Experimentally, "MADDPG + policy mask" performs similarly or worse than MADDPG on all but one environment, and never outperforms the full CoachReg approach. However, policy masks alone seem sufficient to succeed on SPREAD, which is about selecting a landmark from a set. Finally "MADDPG + agent modeling" does not drastically improve on MADDPG apart from one environment and is always outperformed by the full TeamReg (except on COMPROMISE, see Subsection 4.4.2) which supports the importance of enforcing predictability alongside agent modeling.

4.4.2 Effects of enforcing predictable behavior

Improving agent modeling

Here we validate that enforcing predictability makes the agent-modeling task more successful. To this end, we compare, on the SPREAD environment, the team-spirit losses between TeamReg and its ablated versions. Figure 4.7 shows that initially, due to the weight initialization, the predicted and actual actions both have relatively small norms yielding small values of team-spirit loss. As training goes on (∼1000 episodes), the norms of the action vector increase, and the regularization loss becomes more important. As expected, MADDPG leads to the worst team-spirit loss as it is not trained to predict the actions of other agents. When using only the agent-modeling objective ($\lambda_1 > 0$),

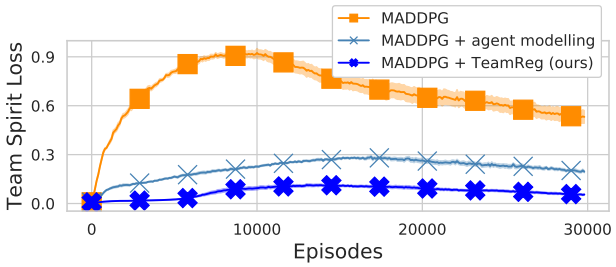


Figure 4.7: Effect of enabling and disabling the coefficients λ_1 and λ_2 on the ability of agents to predict their teammates' behaviors. Solid lines and envelope are average and SEM on 10 seeds on SPREAD.

the agents significantly decrease the team-spirit loss, but it never reaches values as low as when using the full TeamReg objective ($\lambda_1 > 0$ and $\lambda_2 > 0$). Note that the team-spirit loss increases when performance starts to improve i.e. when agents start to master the task (∼8000 episodes). Indeed, once the return maximization signal becomes stronger, the relative importance of the auxiliary objective is reduced. Being predictable with respect to one another may push agents to explore in a more structured and informed manner in the absence of a reward signal, as similarly pursued by intrinsic motivation approaches (Chentanez et al., 2005).

The case of competitive settings

The results presented in Figure 4.6 show that MADDPG + TeamReg is outperformed by all other algorithms when considering average return across agents. In this section, we seek to further investigate this failure mode. Importantly, COMPROMISE is the only task with a competitive component (i.e. the only one in which agents do not share their rewards). The two agents being strapped together, a good policy has both agents reach their landmark successively (e.g. by having both agents navigate towards the closest landmark). However, if one agent never reaches for its landmark, the optimal strategy for the other one becomes to drag it around and always go for its own, leading to a strong imbalance in the return cumulated by both agents. While such a scenario does not occur for the other algorithms, we found TeamReg to often lead to cases of domination such as depicted in Figure 4.9.

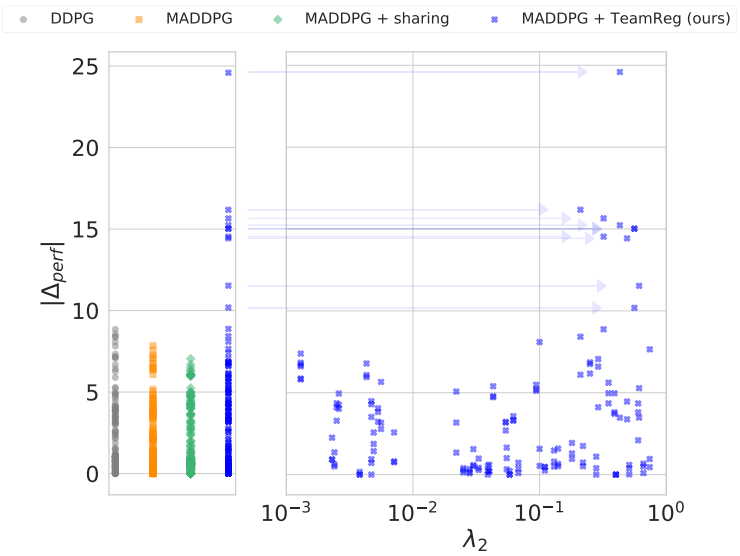


Figure 4.8: Average performance difference (Δ_{perf}) between the two agents in COMPROMISE for each of the 150 runs of the hyperparameter searches (left). All occurrences of abnormally-high performance differences are associated with high values of λ_2 (right).

Figure 4.8 depicts the performance difference between the two agents for every one of the 150 runs of the hyperparameter search for TeamReg and the baselines and shows that (1), TeamReg is the only algorithm that leads to large imbalances in performance between the two agents, and (2), that these cases where one agent becomes dominant are all associated with high values of λ_2, which drives the agents to behave in a predictable fashion to one

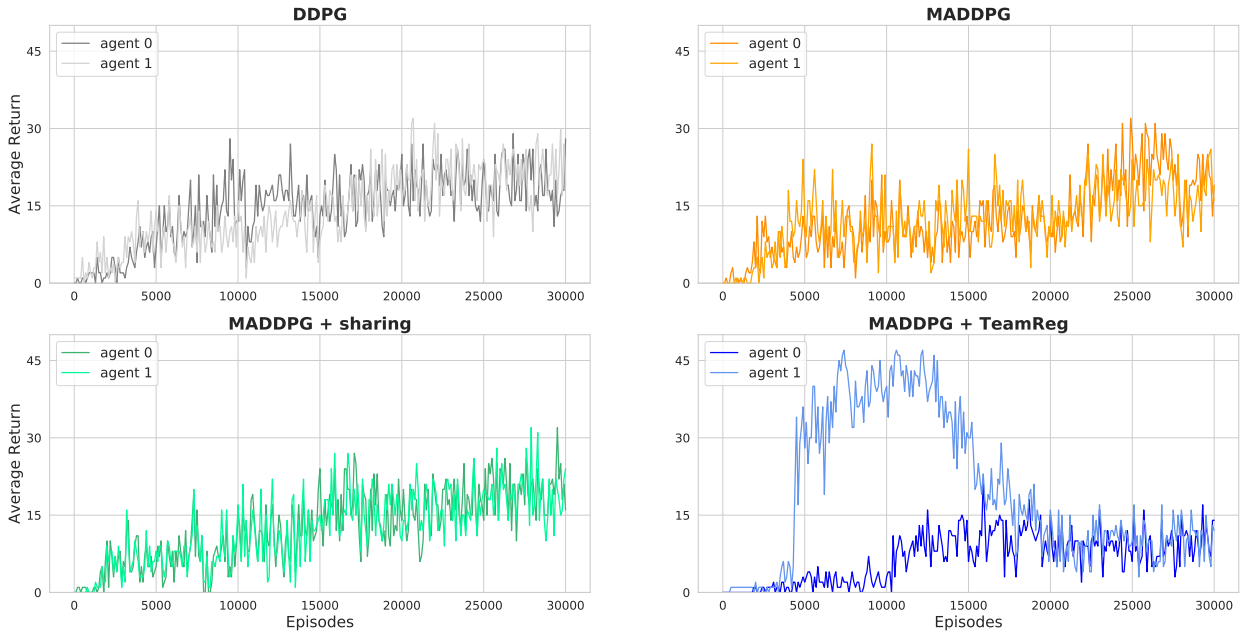


Figure 4.9: Learning curves for TeamReg and the three baselines on COMPROMISE. We see that while both agents remain equally performant as they improve at the task for the baseline algorithms, TeamReg tends to make one agent much stronger than the other one. This domination is optimal as long as the other agent remains docile, as the dominant agent can gather much more reward than if it had to compromise. However, when the dominated agent finally picks up the task, the dominant agent, which has learned a policy that does not compromise, sees its return dramatically go down and the mean over agents overall then remains lower than for the baselines.

another.

Looking back at Figure 4.9, while these domination dynamics tend to occur at the beginning of training, the dominated agent eventually gets exposed more and more to sparse reward gathered by being dragged (by chance) onto its own landmark, picks up the goal of the task and starts pulling in its own direction, which causes the average return over agents to drop as we see happening midway during training in Figure 4.6. These results suggest that using a predictability-based team regularization in a competitive task can be harmful; quite understandably, you might not want to optimize an objective that aims at making your behavior predictable to (and influenceable by) your opponent.

4.4.3 Analysis of synchronous and coherent sub-policy selection

In this section we confirm that CoachReg yields the desired behavior: agents *synchronously* and *coherently* alternating between *varied* sub-policies.

Figure 4.10 shows the average entropy of the mask distributions for each environment compared to the entropy of Categorical Uniform Distribution (CUD) of size k (k-CUD). On all the environments, agents use several masks and tend to alternate between masks with more variety (close to uniformly switching between 3 masks) on SPREAD (where there are 3 agents and 3 goals) than on the other environments (comprised of 2 agents). Moreover, the Hamming proximity between the agents' mask sequences, $1 - D_h$ where D_h is the Hamming distance (i.e. the ratio of timesteps for which the two sequences are different) shows that agents are synchronously selecting the same policy mask at test time (without a coach). Finally, we observe that some settings result in the agents coming up with interpretable strategies, like the one depicted in Figure 4.13 in Subsection 4.4.3 where the agents alternate between two sub-policies depending on the position of the target (see also animations at `https://sites.google.com/view/marl-coordination/`).

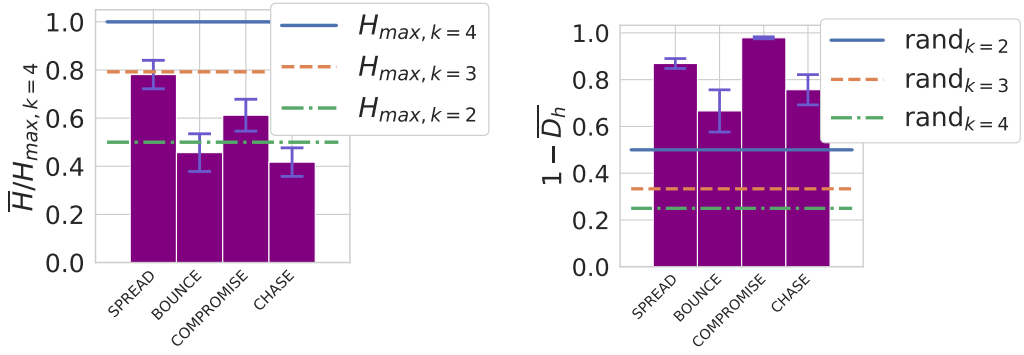


Figure 4.10: (Left) Average entropy of the policy mask distributions for each task. $H_{max,k}$ is the entropy of a k-CUD. (Right) Average Hamming Proximity between the policy mask sequence of agent pairs. rand_k stands for agents independently sampling their masks from k-CUD. Error bars are the SEM on 10 seeds.

Mask diversity and synchronicity (ablation)

As in Subsection 4.4.3 we report the mean entropy of the mask distribution and the mean Hamming proximity for the ablated "MADDPG + policy mask" and compare it to the full CoachReg. With "MADDPG + policy mask" agents are not incentivized to use the same masks.

Therefore, in order to assess if they synchronously change policy masks, we computed, for each agent pair, seed, and environment, the Hamming proximity for every possible masks equivalence (mask 3 of agent 1 corresponds to mask 0 of agent 2, etc.) and selected the equivalence that maximized the Hamming proximity between the two sequences.

We can observe that while "MADDPG + policy mask" agents display a more diverse mask usage, their selection is less synchronized than with CoachReg. This is easily understandable as the coach will tend to reduce diversity in order to have all the agents agree on a common mask, on the other hand, this agreement enables the agents to synchronize their mask selection. To this regard, it should be noted that "MADDPG + policy mask" agents are more synchronized than agents independently sampling their masks from k-CUD, suggesting that, even in the absence of the coach, agents tend to synchronize their mask selection.

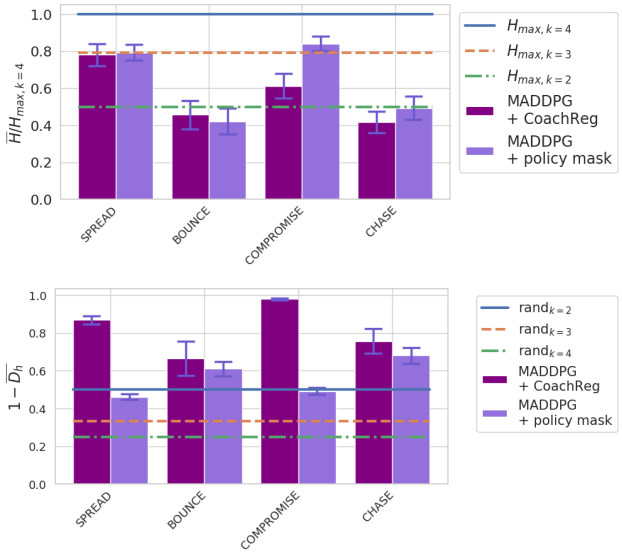


Figure 4.11: (Top) Entropy of the policy mask distributions for each task and method, averaged over agents and training seeds. $H_{max,k}$ is the entropy of a k-CUD. (Bottom) Hamming Proximity between the policy mask sequence of each agent averaged across agent pairs and seeds. rand_k stands for agents independently sampling their masks from k-CUD. Error bars are SEM across seeds.

Mask densities

We depict in Figure 4.12 the mask distribution of each agent for each *(seed, environment)* experiment when collected on 100 different episodes. Firstly, in most of the experiments, agents use at least 2 different masks. Secondly, for a given experiment, agents' distributions are very similar, suggesting that they are using the same masks in the same situations and that they are therefore synchronized. Finally, agents collapse more to using only one mask on CHASE, where they also display more dissimilarity between one another. This may explain why CHASE is the only task where CoachReg does not improve performance. Indeed, on CHASE, agents do not seem synchronized nor leveraging multiple sub-policies which are the priors to coordination behind CoachReg. In brief, we observe that CoachReg is less effective in enforcing those priors to coordination in CHASE, an environment where it does not boost nor harm performance.

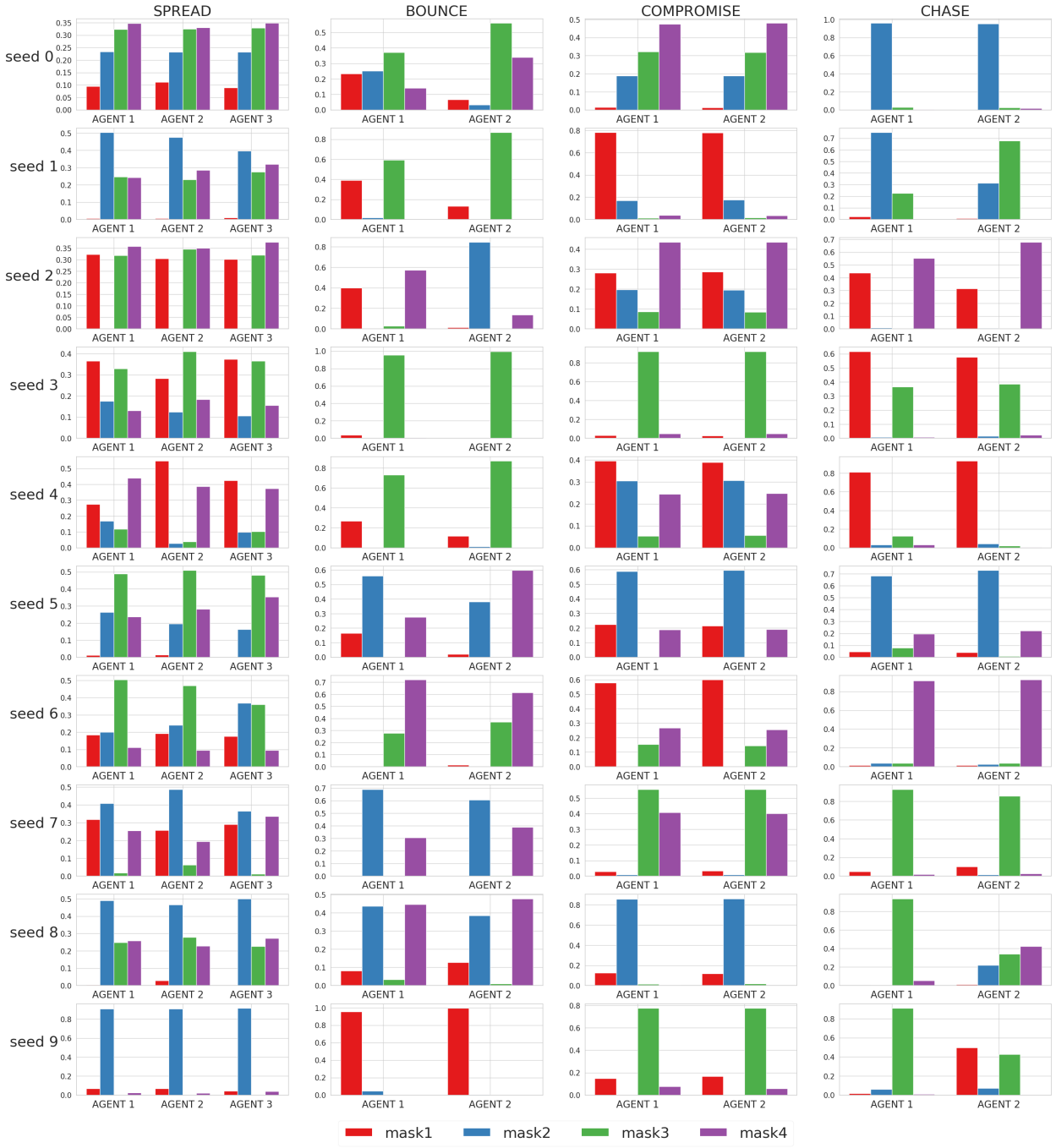
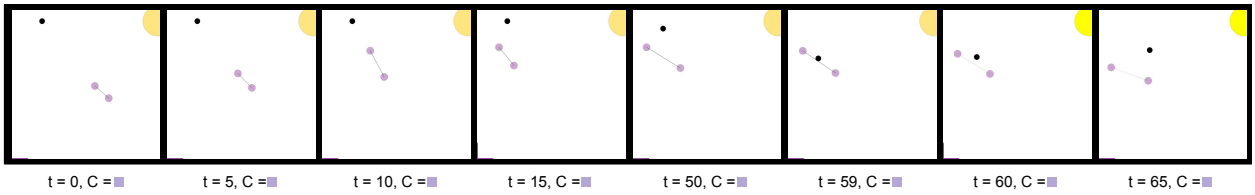


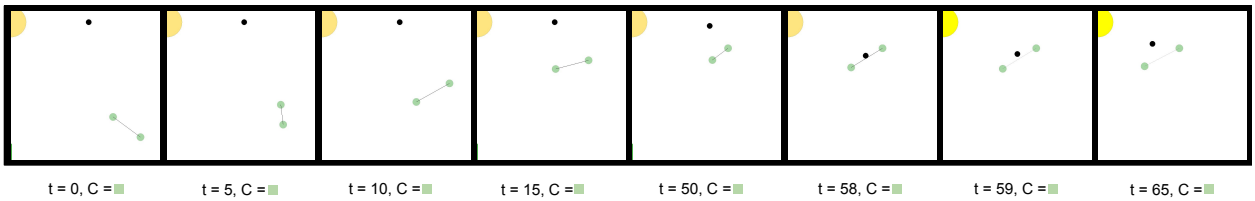
Figure 4.12: Agent's policy mask distributions. For each *(seed, environment)* we collected the masks of each agent on 100 episodes.

Episodes rollouts with synchronous sub-policy selection

We display here and on `https://sites.google.com/view/marl-coordination/` some interesting sub-policy selection strategies evolved by CoachReg agents. In Figure 4.13, the agents identified two different scenarios depending on the target-ball location and use the corresponding policy mask for the whole episode. Whereas in Figure 4.13, the agents synchronously switch between policy masks during an episode. In both cases, the whole group selects the same mask as the one that would have been suggested by the coach.

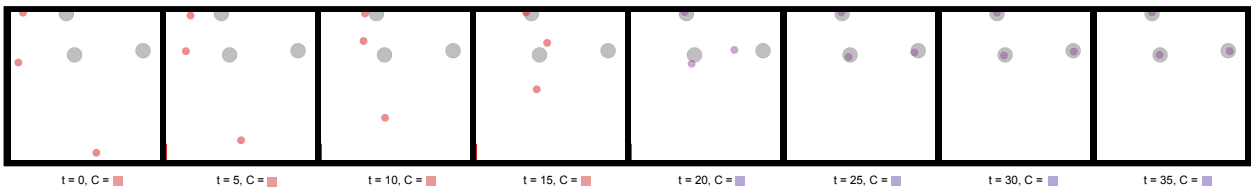


(a) BOUNCE: The ball is on the left side of the target, agents both select the purple policy mask

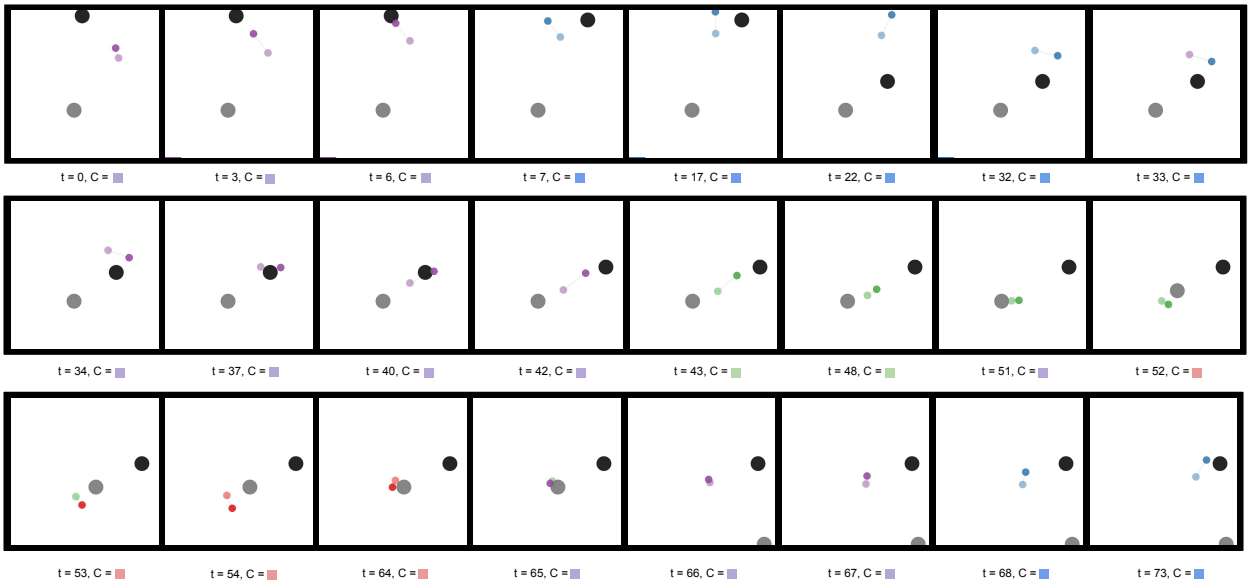


(b) BOUNCE: The ball is on the right side of the target, agents both select the green policy mask

Figure 4.13: Visualization of two different BOUNCE evaluation episodes. Note that here, the agents' colors represent their chosen policy mask. Agents have learned to synchronously identify two distinct situations and act accordingly. The coach's masks (not used at evaluation time) are displayed with the timestep at the bottom of each frame.



(a) SPREAD



(b) COMPROMISE

Figure 4.14: Visualization of sequences on two different environments. An agent's color represents its current policy mask. The coach's masks (not used at evaluation time) are displayed with the timestep at the bottom of each frame. Agents synchronously switch between the available policy masks.

4.4.4 Scalability with the number of agents

Complexity

In this section, we discuss the increases in model complexity that our methods entail. In practice, this complexity is negligible compared to the overall complexity of the CTDE framework. To that respect, note that (1) the critics are not affected by the regularizations, so our approaches only increase complexity for the forward and backward propagation of the actor, which consists of roughly half of an agent's computational load at training time. Moreover, (2) efficient design choices significantly impact real-world scalability and performance: we implement TeamReg by adding only additional heads to the pre-existing actor model (effectively sharing most parameters for the teammates' action predictions with the agent's action selection model). CoachReg consists only of an additional linear layer per agent and a unique Coach entity for the whole team (which scales better than a critic since it only takes observations as inputs). As such, only a small number of additional parameters need to be learned relative to the underlying base CTDE algorithm. For a TeamReg agent, the number of parameters of the actor increases linearly with the number of agents (additional heads) whereas the critic model grows quadratically (since the observation sizes themselves usually depend on the number of agents). In the limit of increasing the number of agents, the proportion of added parameters by TeamReg compared to the increase in parameters of the centralized critic vanishes to zero. On the SPREAD task, for example, training 3 agents with TeamReg increases the number of parameters by about 1.25% (with a similar computational complexity increase). With 100 agents, this increase is only 0.48%. For CoachReg, the increase in an agent's parameter is independent of the number of agents. Finally, any additional heads in TeamReg or the Coach in CoachReg are only used during training and can be safely removed at execution time, reducing the system's computational complexity to that of the base algorithm.

Robustness

To assess how the proposed methods scale to a greater number of agents, we increase the number of agents in the SPREAD task from three to six agents. The results presented in Figure 4.15 show that the performance benefits provided by our methods hold when the number of agents is increased. Unsurprisingly, we also note how quickly learning becomes more challenging when the number of agents rises. Indeed, with each new agent, the coordination problem becomes more and more difficult, and that might explain why our methods that promote coordination maintain a higher degree of performance. Nonetheless, in the sparse reward setting, the complexity of the task soon becomes too difficult and none of the algorithms is able to solve it with six agents.

While these results show that our methods do not contribute to a quicker downfall when the number of agents is increased, they are not however aimed at tackling the problem of

massively multi-agent RL. Other approaches that use attention heads (Iqbal and Sha, 2019) or restrict one agent perceptual field to its n-closest teammates are better suited to these particular challenges and our proposed regularisation schemes could readily be adapted to these methods as well.

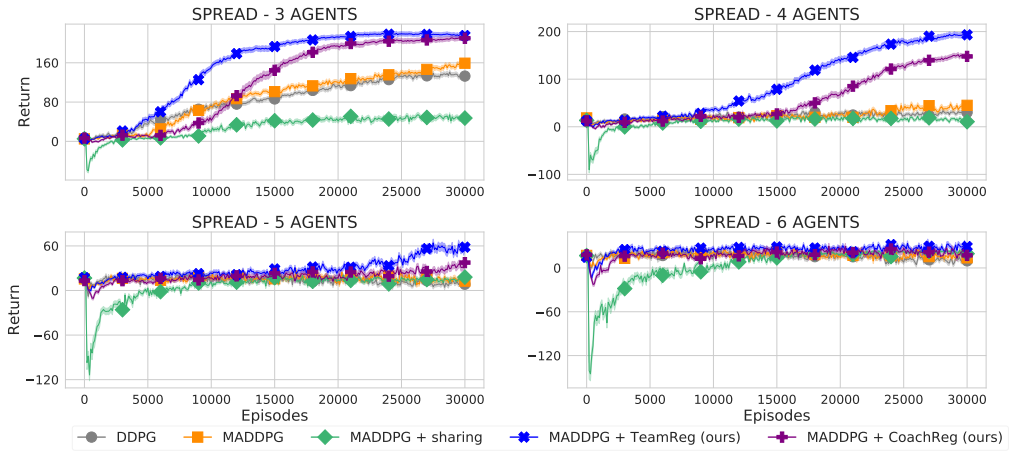


Figure 4.15: Learning curves (mean return over agents) for all algorithms on the SPREAD environment for varying number of agents. Solid lines are the mean and envelopes are the SEM across the 10 training seeds.

4.4.5 Experiments on discrete action spaces

We evaluate our techniques on the more challenging task of 3vs2 Google Research football environment (Kurach et al., 2019). In this environment, each agent controls an offensive player and tries to score against a defensive player and a goalkeeper controlled by the engine's rule-based bots. Here agents have discrete action spaces of size 21, with actions like moving direction, dribble, sprint, short pass, high pass, etc. We use as observations 37-dimensional vectors containing players' and ball's coordinates, directions, etc.

The algorithms presented in Table 4.12 were trained using 25 randomly sampled hyperparameter configurations. The best configuration was retrained using 10 seeds for 80,000 episodes of 100 steps. Table 2 shows the mean return (± standard error across seeds) on the last 10,000 episodes. All algorithms but MADDPG + CoachReg fail to reliably learn policies that achieve positive returns (i.e. scoring goals).

MADDPG	0.004 ± 0.002
+ sharing	0.005 ± 0.003
+ TeamReg (ours)	0.006 ± 0.003
+ CoachReg (ours)	**0.088 ± 0.017**

Table 4.12: Average Returns for 3v2 football

Figure 4.16: Snapshot of the google research football *3vs1-with-keeper*.

4.5 Discussion

TeamReg falls in the line of work that explores how to shape agents' behaviors with respect to other agents through auxiliary tasks. Strouse et al. (2018) use the mutual information between the agent's policy and a goal-independent policy to shape the agent's behavior towards hiding or spelling out its current goal. However, this approach is only applicable to tasks with an explicit goal representation and is not specifically intended for coordination. Jaques et al. (2019) approximate the direct causal effect between agents' actions and use it as an intrinsic reward to encourage social empowerment. This approximation relies on each agent learning a model of other agents' policies to predict its effect on them. In general, this type of behavior prediction is referred to as agent modeling (or opponent modeling) and has been used in previous work to enrich representations (Hernandez-Leal et al., 2019; Hong et al., 2017), to stabilize the learning dynamics (He et al., 2016) or to classify the opponent's play style (Schadd et al., 2007). More details are given in Chapter 2.

With CoachReg, agents learn to unitedly recognize different modes in the environment and adapt by jointly switching their policies. This echoes with the hierarchical RL literature

and in particular with the single agent options framework (Bacon et al., 2017) where the agent switches between different sub-policies – the options – depending on the current state. To encourage cooperation in the multi-agent setting, Ahilan and Dayan (2019) proposed that an agent, the "manager", is extended with the possibility of setting other agents' rewards in order to guide collaboration. CoachReg stems from a similar idea: reaching a consensus is easier with a central entity that can asymmetrically influence the group. Yet, Ahilan and Dayan (2019) guides the group in terms of "ends" (influences through the rewards) whereas CoachReg constrains it in terms of "means" (the group must synchronously switch between different strategies). Hence, the interest of CoachReg does not just lie in training sub-policies (which are obtained here through a simple and novel masking procedure) but rather in co-evolving synchronized sub-policies across multiple agents. Mahajan et al. (2019) also looks at sub-policies co-evolution to tackle the problem of joint exploration, however, their selection mechanism occurs only on the first timestep and requires duplicating random seeds across agents at test time. On the other hand, with CoachReg the sub-policy selection is explicitly decided by the agents themselves at each timestep without requiring a common sampling procedure since the mode recognition has been learned and grounded on the state throughout training.

Barton et al. (2018) propose Convergent Cross Mapping (CCM) to measure the degree of effective coordination between two agents. Although this represents an interesting avenue for behavior analysis, it fails to provide a tool for effectively enforcing coordination as CCM must be computed over long time series making it an impractical learning signal for single-step temporal difference methods.

To our knowledge, our work is the first to extend agent modeling to derive an inductive bias toward team-predictable policies or to introduce a collective, agent-induced, modulation of the policies without an explicit communication channel. Importantly, these coordination proxies are enforced throughout training only, which allows for maintaining decentralized execution at test time.

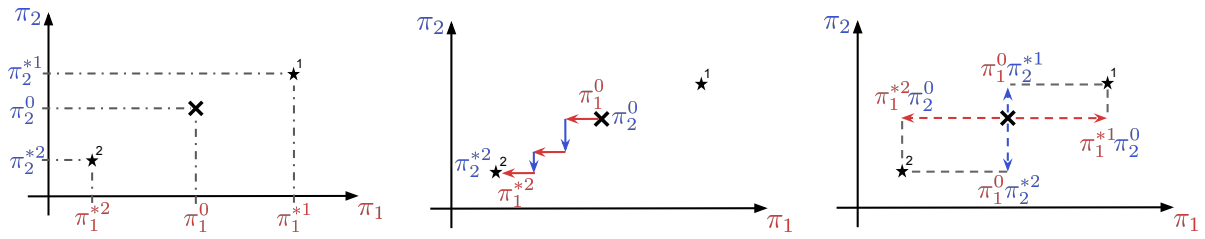
Chapter 5

Coordination in the Offline Setting

Training multiple agents to coordinate is an important problem with applications in robotics, game theory, economics, and social sciences. However, most existing MARL methods are online and thus impractical for real-world applications in which collecting new interactions is costly or dangerous. While these algorithms should leverage offline data when available, doing so gives rise to *the offline coordination problem*. Specifically, we identify and formalize the *Strategy Agreement (SA)* and the *Strategy Fine-Tuning (SFT)* challenges, two coordination issues at which current offline MARL algorithms fail. To address this setback, we propose a simple model-based approach that generates synthetic interaction data and enables agents to converge on a strategy while fine-tuning their policies accordingly. Our resulting method, Model-based Offline Multi-Agent Proximal Policy Optimization (MOMA-PPO), outperforms the prevalent learning methods in challenging offline multi-agent MuJoCo tasks even under severe partial observability and with learned world models. This chapter features work from Barde et al. (2023).

5.1 The offline coordination problem

Multi-agent problems are ubiquitous in real-world scenarios, including traffic control, distributed energy management, multi-robot coordination, auctions and marketplaces, and social networks (Dresner and Stone, 2004; Palanisamy, 2020; Steeb et al., 1981; Boyan and Littman, 1993; Varga et al., 1994; Huang et al., 1995; Brauer and Weiß, 1998; Fischer et al., 1993; Cederman, 1997; Grand et al., 1997; Crawford and Sobel, 1982; Panait and Luke, 2005). This makes the development of efficient multi-agent algorithms a crucial research area in artificial intelligence and machine learning (Vinitsky et al., 2022; Shi et al., 2021; Zhang et al., 2019; Liu and Wu, 2021; Li et al., 2021) with substantial implications in various fields including robotics, game theory, economics, and social sciences. However, existing methods are mostly online and require interactions with the environment throughout learn-



(a) multi-agent policy space (b) online learning (c) offline learning

Figure 5.1: Illustration of the offline coordination problem. (a) Policy space for a task with two optima. (b) In online learning, agents continuously co-evolve and adapt to one another, eventually converging to a globally optimal equilibrium. (c) In offline learning, agents cannot interact to estimate how their current policy might fare or to probe other agents' current behavior. They have to estimate the optimal strategies and the corresponding policies only from the interactions in the offline datasets. Choosing between optimum $\star^1$ and optimum $\star^2$ is referred to as *Strategy Agreement* while *Strategy Fine-Tuning* relates to deriving the corresponding optimal policies.

ing which often makes them costly or even dangerous for real-world applications (Levine et al., 2020). In contrast, offline Reinforcement Learning obviates the need for interactions with the environment as it allows learning from extensive existing datasets that do not have to be collected by experts. It is therefore well suited to tasks for which one cannot afford to materialize the situation in practice, building a simulator is unfeasible, and there exist datasets of realizations of such situations. Consequently, we hypothesize that offline multi-agent approaches will be key for tackling real-world multi-agent problems. Let us imagine for instance trying to understand how autonomous actors (i.e., governments, international organizations, industries, etc.) must maneuver to reduce the severity of a worldwide pandemic while preventing economic collapse. It goes without saying that starting pandemics is not a viable way to gain real-world practice and building a simulator is a colossal task that would suggest emulating our society and its economy. Yet, the impact of past decisions (such as implementing lockdown policies, travel restrictions, and vaccination campaigns) on the unfolding of the pandemics and the economy is well-documented. These records could be used to derive new strategies in the future.

In general, actors such as individuals, organizations, robots, software processes, cars, etc., are self-governed and ultimately act autonomously. However, during the offline learning phase, it is reasonable to assume that learners can share pieces of information, which makes the Centralized Training and Decentralized Execution formulation a natural approach for the offline training regime. Unfortunately, as shown in this work, simply combining CTDE MARL and offline RL methods does not ensure that agents learn policies which perform well together from a dataset of multi-agent interactions. Indeed, multi-agent problems require agents to coordinate, that is to act coherently as a group such that individual behaviors combine into an efficient team policy.

In this work, we propose that multi-agent coordination can be decomposed into two distinct challenges. First, since most multi-agent tasks allow multiple optimal team strategies (Boutilier, 1996b) – some of which may only vary on how they break symmetries present in the task (Hu et al., 2020) – agents must select, as a group, one strategy over another so that they individually converge toward coherent behaviors. We refer to this coordination challenge as *Strategy Agreement*. Additionally, for a chosen team strategy, agents have to precisely calibrate and adjust their behaviors to one another if they are to rise toward the corresponding optimal group behavior. We call this *Strategy Fine-Tuning*. In online learning, agents continuously interact together in the environment, therefore, changes due to local optimization directly impact other agents and teammates are able to adapt: coordination occurs through interactive trial and error. Conversely, when learning from a fixed dataset of interactions, agents cannot probe how other agents are adapting their behaviors, which global strategy they may be choosing, or how the learned individual policies might combine and perform together in the environment. Thus, it is difficult for offline learners to coordinate. Figure 5.1 illustrates this in the policy space: (a) shows a task for which there exist two optimal team strategies. (b) in online learning agents continuously interact and reach a global optimum by co-adapting and improving on each other's changes. (c) in the offline setting however, agents must first independently decide towards which of the two optima they aim to converge (i.e., Strategy Agreement between π_i^{*1} or π_i^{*2}, $i = 1, 2$), let us assume they pick $\star 1$. Then, they must derive their corresponding optimal policy (i.e., Strategy Fine-Tuning toward π_i^{*1}, $i = 1, 2$) without additional interactions that inform them how their current policy behaves in the environment or blend with the other agent's policy (they only have access to interactions with dataset policies π_i^0, $i = 1, 2$).

Current methods (Yang et al., 2021; Jiang and Lu, 2021; Pan et al., 2022) deal with offline MARL by simply extending single-agent offline RL to the multi-agent setting. To do so, they either consider that agents are independent learners, or leverage the CTDE paradigm. In such settings, it is possible in theory for the agents to learn (provided that the datasets have enough coverage) the different optimal policies, yet, since agents are never actually evaluated together during training, they fail to agree on which strategy to pick and how to fine-tune their behavior to one another. We illustrate such offline coordination failure in an offline version of the well-established Iterated Coordination Game (Boutilier, 1996b). Thus, we motivate the need for generating additional synthetic data in order for agents to assess how they would interact and eventually allow them to coordinate. We propose MOMA-PPO, a simple model-based approach to generate such synthetic interactions and show that it allows for offline coordination. We extend our solution to more complex tasks with partial observability in Multi-Agent MuJoCo (MAMuJoCo) environments (Peng et al., 2021) and show that it learns effective strategies that outperform the offline MARL baselines. Interestingly, MOMA-PPO also outperforms the fully centralized model-free method IQL (Kostrikov et al., 2021) even though centralization completely bypasses the strategy agreement problem. This suggests that model-free methods, even when fully centralized, are

unable to deal with strategy fine-tuning. On the other hand, our model-based approach fixes both multi-agent strategy agreement and strategy fine-tuning problems. This goes to show that the benefits of model-based approaches over model-free ones (Yu et al., 2020) hold in the offline multi-agent setting. Finally, we observe in our experiments that the single-agent approach of fine-tuning offline methods (i.e. IQL) with online interactions (Kostrikov et al., 2021) fails for multi-agent formulation (i.e. MAIQL). Conversely, MOMA-PPO's success puts forward the benefits of model-based approaches that leverage online policy learning succeeds.

5.2 A model-based solution to the offline coordination problem

In this work, we propose MOMA-PPO, a Dyna-like (Sutton, 1990) model-based approach to multi-agent CTDE offline learning that relies on PPO (Schulman et al., 2017). The method can be decomposed into two steps: first learning a world model from the dataset, and then using the world model to train the agents' policies.

5.2.1 Learning a centralized world model ensemble

MOMA-PPO leverages the CTDE assumptions and therefore learns centralized models to predict the next state, reward, and termination condition from the current state and actions. When learning in an approximate world model, a risk is that RL agents learn to exploit the world model's reconstruction inaccuracies to reap more rewards in simulation, eventually producing incoherent behaviors that perform poorly in the real world (Ha and Schmidhuber, 2018). One way to avoid this is to penalize the agent for going into regions of the state-action space where the world model is uncertain about its predictions (Yu et al., 2020). Learning an ensemble of models enables estimating the world model's epistemic uncertainty due to the limited amount of learning data in the offline dataset. Each model comprises two diagonal Gaussians $\mathcal{N}(\mu_T, \sigma_T^2)$ and $\mathcal{N}(\mu_r, \sigma_r^2)$ that respectively model the next state s' and the reward r. Each model also predicts whether or not the next state is terminal using a Bernoulli distribution $\text{Bern}(p_d)$. μ_T, μ_R, σ_T, σ_R, and p_d are parametrized by neural networks conditioned on the current global state s and the joint action a. The parameters are trained using Gaussian negative log-likelihood – for $\mathcal{N}(\mu_T, \sigma_T^2)$ and $\mathcal{N}(\mu_r, \sigma_r^2)$ – and binary cross-entropy – for $\text{Bern}(p_d)$ – on the offline dataset $\mathcal{D}$. In practice, we train $N_m = 7$ models and keep the best $N = 5$ based on their average validation accuracy across the next states and rewards. We estimate the epistemic uncertainty of the reward using the variance of the

predicted rewards across the ensemble:

$$\epsilon_r = \frac{\sum_{m=1}^{N}(\hat{r}_m - \bar{r})^2}{N-1}, \qquad \bar{r} = \frac{\sum_{m=1}^{N}\hat{r}_m}{N}.$$

We also estimate the epistemic uncertainty of the general prediction by concatenating the next state and the reward and computing the Frobenius norm of the ensemble covariance matrix:

$$\epsilon_g = ||\text{cov}(x_i, x_j)||_F \quad \text{with} \quad \text{cov}(x_i, x_j) = \frac{\sum_{m=1}^{N}(\hat{x}_{i,m} - \bar{x_j})(\hat{x}_{j,m} - \bar{x_i})}{N-1},$$

where x_i and x_j are components of the vector resulting from the concatenation of the predicted next state vector $\hat{s}'$ and the predicted reward scalar $\hat{r}$.

At this point, we define a world model based on the ensemble such that:

$$\hat{s}_{t+1}, \bar{r}_t, \bar{f}_t, \epsilon_{r,t}, \epsilon_{g,t} \sim \mathcal{M}(\cdot|s_t, a_t),$$

where $\bar{f}_t$ is a mask equal to 0, if the model predicts that we reached an absorbing state, and 1 otherwise. $\bar{r}_t$ is the mean predicted reward across the ensemble and $\bar{f}_t$ results from a majority vote between the members of the ensemble. Since the mean state would likely be out-of-distribution and lack the structure of real states (which would impede learning and evaluation), $\hat{s}_{t+1}$ is instead sampled uniformly amongst the possible predicted next states from the ensemble. Finally, to avoid unrealistic values, $\bar{r}_t$ and $\hat{s}_{t+1}$ are clipped to the minimum bounding box of the offline dataset while uncertainties estimations are limited to a specified threshold:

$$\min_{r\in\mathcal{D}} r \leq \bar{r}_t \leq \max_{r\in\mathcal{D}} r, \quad \min_{s_i\in\mathcal{D}} s_i \leq \hat{s}_{t+1,i} \leq \max_{s_i\in\mathcal{D}} s_i \quad \forall i \in [0, q] \,|\, \mathcal{S} \subset \mathbb{R}^q, \quad \text{and} \quad \epsilon_r \leq l_\epsilon,\ \epsilon_g \leq l_\epsilon.$$

5.2.2 Model-based Offline Multi-Agent Proximal Policy Optimization (MOMA-PPO)

Once $\mathcal{M}$ has been trained on the offline dataset $\mathcal{D}$, it can be used to train online reinforcement learning algorithms in a Dyna-like manner. In this work, we decide to use MAPPO, a CTDE multi-agent version of PPO (Yu et al., 2022).

The synthetic data used to train the PPO policies is collected by sampling states from the offline dataset $\mathcal{D}$ and using the current policies π^i alongside the world model $\mathcal{M}$ to generate PPO's training rollouts of size k. Terminating a rollout when the world model uncertainty exceeds l_ϵ allows for adaptative rollout length and avoids training the policies on unfeasible

data. Rollouts are always terminated with a timeout mark ζ_t such that:

$$\zeta_t = 1 - \mathbb{I}(t = k - 1 \cup \epsilon_{g,t} \geq l_\epsilon),$$

where $\mathbb{I}$ is the indicator function. On the PPO side, we adapt the Generalized Advantage Estimator (GAE) (Schulman et al., 2015b) to account for these timeouts and ensure that there is no accumulation across rollouts while computing returns. We use the value of the last state as an estimation of the return-to-go (for more detail see Subsection 5.6.1).

The generation of length $k = 3$ rollouts is illustrated in Figure 5.2 where it can be seen that a rollout is interrupted early because the world model was unreliable when generating the associated state (shown in red).

In addition to the adaptive rollout length, the model epistemic uncertainty is used to penalize the agents and avoid exploiting the world model in poorly reconstructed regions of the state space. The final uncertainty-penalized reward is given by:

$$\tilde{r}_t = \bar{r}_t - \lambda_r \epsilon_r - \lambda_g \epsilon_g,$$

where ϵ_r and ϵ_g are hyperparameters that weigh the severity of the penalty.

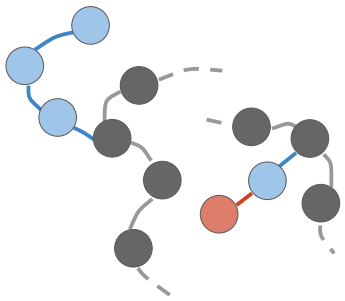


Figure 5.2: Model-based rollouts generation (blue) from dataset's states (grey). Red denotes early termination and $k = 3$.

Practical considerations.

Note that, we assume CTDE and that the global state s_t fully observes the environment, therefore we do not equip the world model with memory. The agents, on the other hand, only have access to partial observations and must rely on action-observation histories h_t^i. In practice, we restrict action-observation histories to 10 steps in the past (either from the dataset or the generated rollouts) and process them with one layer of self-attention (Vaswani et al., 2017) followed by one layer of soft-attention (Bahdanau et al., 2015). The resulting embeddings are concatenated to the agent's current state, and for simplicity, we abuse notation by denoting this "memory enhanced" state by h_t^i.

Our MAPPO implementation uses the QMIX value-decomposition (Rashid et al., 2018) for the centralized value function. All models are kept simple: two layers MLPs for actors and critics and four layers MLPs for world models.

Finally, it is important to note that the task of the MOMA-PPO agents is quite different from the task of the agents that generated the dataset. First, the MOMA-PPO agents' initial state distribution is now the dataset's state distribution. Then the reward of the task has been altered to account for the model uncertainty. Last but not least, agents are only allowed to stray k steps away from the dataset's coverage. While this restriction mitigates world model abuse, it can also prevent the agents from discovering goals that are further away

from the offline data. Our resulting model-based offline multi-agent method is illustrated in Algorithm 3 and more details are provided in Section 5.6.

Algorithm 3 MOMA-PPO

Require: offline dataset $\mathcal{D}$, world model $\mathcal{M}$, rollout horizon k, rollout batch size b, uncertainty penalty coefficients λ_r and λ_g, uncertainty threshold l_ϵ, MAPPO agents.

Initialize MAPPO policies π^i and value function V.
for epoch $1, 2, \ldots$ **do**
  ▷ Generate synthetic data
  Initialize an empty rollout buffer $\mathcal{R} \leftarrow \varnothing$.
  for $1, 2, \ldots, b$ (in parallel) **do**
    Sample history $h_t = \{h_t^i\}_{i=1}^{N_A}$ from $\mathcal{D}$.
    for $j = t, t+1, \ldots, t+k-1$ **do**
      Sample $a_j^i \sim \pi^i(a^i|h_j^i)\, \forall i$.
      Sample $\hat{s}_{j+1}, \bar{r}_j, \bar{f}_j, \epsilon_{r,j}, \epsilon_{g,j} \sim \mathcal{M}(\cdot|s_j, a_j)$.
      Compute $\tilde{r}_j = \bar{r}_j - \lambda_r \epsilon_r - \lambda_g \epsilon_g$.
      Compute $\zeta_j = 1 - \mathbb{I}(j = t+k-1 \cup \epsilon_{g,j} \geq l_\epsilon)$
      Add sample $(h_j, a_j, \tilde{r}_j, \hat{f}_j, \zeta_j, \hat{s}_{j+1})$ to $\mathcal{R}$.
      Get h_{j+1}^i from h_j^i, a_j^i and $\hat{s}_{j+1}$.
      Set $s_{j+1} = \hat{s}_{j+1}$.
  ▷ Train agents
  Use synthetics rollouts in $\mathcal{R}$ to train policies π^i and value function V with MAPPO.
return multi-agent policies π^i.

5.3 Baselines, Environments, Tasks, and Datasets

5.3.1 Baselines

We compare with a large and varied array of baselines. First, we consider a simplified version of the offline MARL problem by assuming centralized training *and centralized execution* with access to the global state. We use IQL (Kostrikov et al., 2021), a state-of-the-art single-agent model-free offline RL algorithm, to tackle this setting. Considering centralized execution gives an upper bound on what can be achieved in terms of strategy agreement. Indeed, a single learner controls all the agents and can thus choose for the whole team the strategy to adopt. Then, we extend IQL to the multi-agent setting by using the QMIX value decomposition on the Q and V value networks. This gives MAIQL, a very competitive model-free CTDE offline MARL algorithm that allows for fine-tuning with additional online data after training (Kostrikov et al., 2021). We refer to the finetuned version as MAIQL-ft

and follow the procedure of Kostrikov et al. (2021): MAIQL-ft is first trained to convergence on the offline data and then finetuned for the same number of training steps by progressively introducing additional interaction data. At the end of finetuning, the replay buffer contains as many offline interactions as online ones.

Recent literature in model-free MARL (de Witt et al., 2020; Lyu et al., 2021) and notably in the offline setting (Pan et al., 2022), advocates for decentralized value functions and independent learners. Therefore, we consider several independent learner approaches, starting with Independent Behavioral Cloning (IBC). Despite its simplicity, BC (Pomerleau, 1991) produces surprisingly efficient baselines for Imitation Learning and offline RL (Barde et al., 2020; Spencer et al., 2021). Finally, we consider the independent learners extensions to Kumar et al. (2020) and Fujimoto and Gu (2021), respectively ITD3+BC and ICQL. We also compare to the state-of-the-art model-free offline MARL method, OMAR (Pan et al., 2022) that we denote IOMAR the highlight that it uses independent learners.

5.3.2 Offline Iterated Coordination Game

To illustrate the strategy agreement coordination challenge, we propose an offline version of the Iterated Coordination Game presented in Figure 5.1 (a). Agents must pick the same direction in order to succeed and we investigate three offline datasets of interactions: in the most favorable one, data is collected by coordinated agents that select the same option of going right most of the time. In the less favorable setting, agent 1 goes left most of the time while agent 2 is more likely to go right. In the neutral setting, agents act uniformly. Table 5.1 (b) reports the policies used to collect the datasets and the resulting datasets' average scores. In the favorable dataset, agents are mostly coordinating (62.3% of the time) while it is the opposite for the unfavorable dataset (37.5% of the time).

		a^2	
		$\leftarrow$	$\rightarrow$
a^1	$\leftarrow$	1,1	0,0
	$\rightarrow$	0,0	1,1

(a)

	$P(a^1 = \rightarrow)$	$P(a^2 = \rightarrow)$	Avg. Score
favorable	0.75	0.75	0.623
neutral	0.5	0.5	0.502
unfavorable	0.25	0.75	0.375

(b)

Table 5.1: Offline Iterated Coordination Game. (a) Pay off matrix of Boutilier (1996b)'s Coordination Game. (b) Datasets' agent policies and corresponding averaged scores

However, all the datasets do contain both coordinated and uncoordinated behaviors in which agents simultaneously choose the same – respectively, different – directions. It is therefore straightforward for a centralized critic to learn that $Q(\rightarrow, \rightarrow) = Q(\leftarrow, \leftarrow) = 1$ while $Q(\rightarrow, \leftarrow) = Q(\leftarrow, \rightarrow) = 0$ on all of the datasets. Yet, decentralized actors remain

unaware of whether they should go left or right since they cannot infer what the other agent will do.

5.3.3 Offline MAMuJoCo

Building upon D4RL (Fu et al., 2020) and MAMuJoCo (Peng et al., 2021), we propose two offline multi-agent continuous control tasks with full and partial observability.

Two-agent Reacher with a mixture-of-expert dataset

To investigate strategy agreement in a more complex continuous control setting, we propose a two-agent version of the Reacher environment as shown in Figure 5.3. The offline dataset is collected as follows: in the first stage, we train online MAPPO on the fully observable two-agent Reacher task (every agent observes all the joint angles and velocities as well as the target position – in black – and the target to fingertip – in green – vector). Depending on the seed of the run, teams converge to counter-clockwise ($\theta_2 \geq 0$ as in Figure 5.3) or clockwise ($\theta_2 \leq 0$) arm bends. Thus we can build a mixture-of-expert dataset by combining equal proportions of demonstrations from clockwise and counter-clockwise teams. Finally, we explore the impact of Full Observability (FO) versus Partial Observability (PO) by considering three types of observation functions: *all-observant* (FO: every agent fully observes the environment), *independent* (PO: each agent only sees the target and the velocity and angle of the joint it controls), and *leader-only* (PO: both agents observes the two joints but only the red agent observes the target's position). Note that with PO no agent observes the target to fingertip vector. These tasks are very challenging and require agents to agree to follow a specific convention (either clockwise or counter-clockwise arm bend) to reach a given target location. Indeed, at least one agent in the team cannot estimate whether or not the fingertip matches the target (they miss information about the other joint or the target).

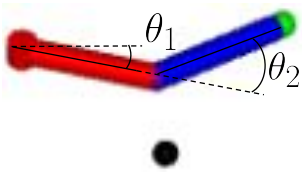


Figure 5.3: Two-agent Reacher. Red and blue agents respectively control the torque on θ_1 and θ_2.

Four-agent Ant

Similarly, we use a MAMuJoCo-like decomposition of the offline ant task to make it multi-agent: each individual limb (composed of two joints) is controlled by a different agent. For the offline datasets, we use the single-agent D4RL datasets and consider two types of observation functions: for fully observable tasks, every agent observes the whole robot (torso observations – i.e., vertical position, orientation, angular and translational velocities – and the observations of all the limbs – i.e., angle and angular velocity of each joint). For the partially observable tasks, agents only observe the limb they control, and the torso observations are only made available to the yellow limb agent. PO tasks are very challenging in this case because only the yellow agent knows if the ant is moving in the correct direction and it must therefore learn to "steer" the whole robot.

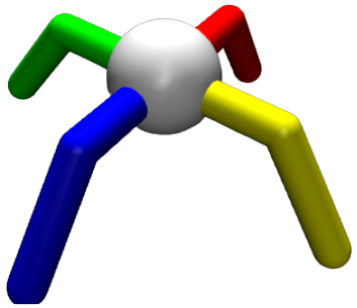


Figure 5.4: Four-agent Ant. Each agent controls a different limb (shown with different colors). In PO tasks, agents only observe the limb they control while the torso – in white – observations are available only to the yellow agent.

MAMuJoCo datasets

Table 5.2 summarizes the datasets' scores distribution as well as the reference performances used to normalize the results. Note that we generated the two-agent reacher mixture-of-expert dataset while ant datasets are from D4RL (Fu et al., 2020) (single agent datasets that we split into multi-agent observations).

Datasets \ Metrics		min (%)	mean (%)	median (%)	max (%)	expert score	random score
reacher	expert-mix	38.8	100.0	98.9	152.3	-4.237	-11.145
ant	random	-3	6.4	7.2	10.3	3879.7	-325.6
	medium	-4.8	80.2	95.1	107.2		
	full-replay	-22.4	72.0	77.8	134.3		
	expert	-32.8	117.4	129.4	142.5		

Table 5.2: Normalized measures of datasets' scores distributions and normalization scores.

5.4 Aggregated results and insights

The experimental procedure such as hyperparameters, training routine, and learning curves are detailed in Section 5.6 for reproducibility. All algorithms are trained to convergence and we used 10 seeds for the Iterated Coordination Game and 3 seeds for MAMuJoCo tasks. Tables are normalized and report the mean evaluation performance and the standard error of the mean across seeds. Evaluation is done for 100 episodes using the greedy policies (no sampling). This section focuses on extracting insights from the experiments results while we provide raw results in Sections 5.7 and 5.6.

5.4.1 Strategy Agreement

Table 5.3 reports the results for the offline Iterated Coordination Game and validates most of our intuitions about strategy agreement: the centralized execution (IQL) and model-based (MOMA-PPO) approaches are able to coordinate agents regardless of the datasets. On the other hand, independent BC agents imitate the dataset behavior and therefore coordinate only if the dataset majorly demonstrates coordination. Surprisingly, the CTDE model-free approach MAIQL is able to break symmetry and coordinate agents in the neutral dataset. We hypothesize that small numerical errors in the centralized value approximation have the team favor one equivalent strategy over the other. Unfortunately, the conservatism of model-free methods forces agents to stay close to the demonstrated behaviors and prevails over this brittle symmetry-breaking mechanism in the unfavorable – i.e., uncoordinated – dataset.

	IQL	MAIQL	IBC	MOMA-PPO
fav.	**1. ± 0.**	**1. ± 0.**	**1. ± 0.**	**1. ± 0.**
neutral	**1. ± 0.**	**0.9 ± 0.1**	0.55 ± 0.11	**1. ± 0.**
unfav.	**1. ± 0.**	0. ± 0.	0. ± 0.	**1. ± 0.**

Table 5.3: Teams' performances on the Coordination Game. MOMA-PPO is the only decentralized execution method to solve it for all the datasets.

Tasks \ Algorithms		centralized	decentralized execution	independent learners				model-based (ours)
		IQL	MAIQL	IBC	ITD3+BC	ICQL	IOMAR	**MOMA-PPO**
FO	all-observant	**1.07 ± 0.01**	0.96 ± 0.05	1.02 ± 0.01	0.78 ± 0.00	0.48 ± 0.06	0.73 ± 0.01	**1.07 ± 0.01**
PO	independent		**0.92 ± 0.04**	0.76 ± 0.04	0.30 ± 0.11	0.46 ± 0.04	0.45 ± 0.02	**0.95 ± 0.06**
	leader-only		0.80 ± 0.05	0.84 ± 0.02	0.48 ± 0.04	0.31 ± 0.05	0.39 ± 0.02	**1.00 ± 0.01**

Table 5.4: Teams' performances on two-agent Reacher with mixture-of-experts dataset for different observation functions. Scores are normalized with expert and random performances. Independent learners fail at strategy agreement on datasets that contain a mixture of experts while MOMA-PPO (and to some extent MAIQL) are able to coordinate agents and match expert performance.

Table 5.4 confirms that these insights on strategy agreement hold in the more complex two-agent Reacher environment. Results also highlight that model-free independent learners

such as IBC, ITD3+BC, ICQL, and IOMAR struggle with strategy agreement – especially under partial observability – and are outperformed by the CTDE method MAIQL.[1] Interestingly, IBC fares best among independent learners. Finally, our model-based CTDE approach *MOMA-PPO is well suited for Strategy Agreement* as it performs on par with centralized execution, a setting that sidesteps the strategy agreement issue altogether. MOMA-PPO also matches or outperforms all other baselines.

5.4.2 Strategy Fine-Tuning

Tasks \ Algorithms		centralized	decentralized execution		independent learners				model-based (ours)
		IQL	MAIQL	MAIQL-ft	IBC	ITD3+BC	ICQL	IOMAR	**MOMA-PPO**
FO	ant-random	0.12 ± 0.00	0.28 ± 0.01	0.28 ± 0.03	0.31 ± 0.00	0.22 ± 0.02	0.08 ± 0.00	0.08 ± 0.00	**0.52 ± 0.07**
	ant-medium	0.97 ± 0.02	0.85 ± 0.02	0.81 ± 0.02	0.84 ± 0.01	1.04 ± 0.00	0.88 ± 0.12	1.10 ± 0.03	**1.29 ± 0.06**
	ant-full-replay	1.22 ± 0.02	0.77 ± 0.21	0.95 ± 0.13	1.20 ± 0.01	1.33 ± 0.01	1.21 ± 0.02	1.30 ± 0.00	**1.42 ± 0.07**
	ant-expert	1.26 ± 0.01	1.24 ± 0.00	1.06 ± 0.07	1.24 ± 0.00	1.25 ± 0.02	0.73 ± 0.15	1.16 ± 0.01	**1.49 ± 0.01**
PO	ant-random		0.31 ± 0.00	0.34 ± 0.04	0.31 ± 0.00	0.31 ± 0.00	0.17 ± 0.02	0.21 ± 0.02	**0.42 ± 0.05**
	ant-medium		0.14 ± 0.02	0.11 ± 0.01	0.17 ± 0.01	0.22 ± 0.05	0.09 ± 0.02	0.06 ± 0.01	**0.54 ± 0.19**
	ant-full-replay		0.18 ± 0.02	-0.07 ± 0.10	0.21 ± 0.02	0.20 ± 0.01	0.09 ± 0.01	0.11 ± 0.02	**0.46 ± 0.10**
	ant-expert		-0.16 ± 0.01	-0.23 ± 0.02	0.05 ± 0.04	0.16 ± 0.00	0.11 ± 0.03	0.10 ± 0.01	**0.18 ± 0.00**

Table 5.5: Teams' performances on four-agent Ant for different datasets and observation functions. Scores are normalized with expert and random performances. Current model-free methods are unable to adapt agents' behaviors and handle partially observable scenarios while MOMA-PPO can learn robust teams.

From Table 5.5 one can investigate how the different offline methods fare with strategy finetuning. First, IQL performs on par with the other model-free methods which suggests that centralized execution (single-agent) vs. decentralized execution (multi-agent) is less a consideration for strategy fine-tuning than it is for strategy agreement. Yet, this also highlights that *model-free methods (even when centralized) are unable to perform Strategy Fine-Tuning.* Indeed, they are surpassed by our model-based method, MOMA-PPO, which generates additional synthetic experiences that *allow for Strategy Fine-Tuning in addition to Strategy Agreement.* For model-free methods, independent learners tend to outperform CTDE ones (which echoes Pan et al. (2022) observations). Additionally, comparing IQL with MAIQL performance does highlight that offline multi-agent coordination is more challenging in varied datasets – such as medium or full-replay that display both coordinated and uncoordinated behaviors – than in uniform and coordinated datasets – such as expert's. Despite its simplicity and the fact that it only does imitation, IBC does surprisingly well as an offline MARL baseline.

In partially observable (PO) tasks, model-free methods are unable to adapt the behaviors demonstrated in the datasets and they result in teams that run in circles because the yellow

[1] `https://sites.google.com/view/moma-ppo` shows independent learners converge to incompatible conventions.

agent (the only one to observe the torso's headings and velocities) fails to correct the other limbs' motions. Conversely, with MOMA-PPO, the yellow agent steers the ant toward the correct direction, and the teams reach very satisfactory performances provided that the datasets have enough coverage to learn a wold-model that can simulate diverse and robust behaviors (cf. the lower performance in the expert dataset).[2]

Finally, the poor performance of MAIQL-ft suggests that IQL's finetuning abilities might not carry over to the multi-agent setting (even though we used the ground-truth simulator to generate the rollouts). While there might be multiple causes, we hypothesize that it is mainly due to MAIQL's instability since it required intensive hyperparameters finetuning and filtering out collapsed runs for ant tasks. We believe that its instability is exacerbated by the induced non-stationarity of the training data when augmenting it with online interactions. Unlike online methods, offline algorithms are designed to learn on fixed datasets and are thus ill-equipped to deal with data continually collected by changing policies. Similarly, experiments that used MAIQL instead of MAPPO for MOMA (i.e. MOMA-IQL) quickly led to unstable learning and exploding losses.

5.4.3 Ablations

Subsection 5.7.2 reports the raw results for the ablations and we discuss here the main insights. First, using the ground-truth simulator yielded higher scores suggesting that the performance of MOMA-PPO can be further improved by learning a more accurate world model. Then, not clipping the generated next-states to the dataset's bounding box prevented generating length 10 rollouts as the states' magnitude exploded after just a few world-model steps. Additionally, the use of uncertainty penalty (λ_g) and adaptive rollouts' length (l_ϵ) are necessary to achieve satisfactory results. Finally, varying the rollout's maximum length from 5 to 50 did not significantly impact performance.

5.5 Discussion and Conclusion

This section concludes by discussing this work in relationship with the literature.

Multi-agent coordination

Coordination has been a challenge of interest since the early works on cooperative MARL (Boutilier, 1996b; Claus and Boutilier, 1998; Littman et al., 2001; Brafman and Tennenholtz, 2002; Chalkiadakis and Boutilier, 2003) and has consistently been a central focus of the multi-agent literature (Zhang and Lesser, 2013; Lowe et al., 2017; Jaques et al., 2019; Lerer and Peysakhovich, 2019). While different works consider different aspects of coordination

[2] `https://sites.google.com/view/moma-ppo` shows rollouts with and without "steering" behavior.

– such as behavior predictability and synchronous sub-policy selection (Roy et al., 2020), structured team exploration (Mahajan et al., 2019) or the emergence of communication and cooperative guiding (Lazaridou et al., 2017; Woodward et al., 2020; Barde et al., 2022) – our definition of coordination is closest to the seminal work of Boutilier (1996b). Indeed, we consider coordination in terms of agents agreeing to individually follow the same team strategy (that is a policy over joint actions) and finetuning their behavior to one another in tasks where multiple distinct optimal team strategies exist. A similar notion of coordination has been used in the *Zero-Shot Coordination problem* investigated by Hu et al. (2020) where agents are trained so that they are able to perform with agents they have never seen before. Yet, while their focus is on deriving standardized coordinated strategies that can generalize to unseen teammates, coordination is still learned through online interactions.

Coordination with teammates without direct interactions is often referred to as *ad-hoc coordination* (Stone et al., 2010; Barrett et al., 2011). Recent works assume access to offline demonstrations of the teammates' behaviors. These can be used to guide the agent's self-play training toward adopting the appropriate equilibrium (or "social conventions") of its future teammates (Lerer and Peysakhovich, 2019; Tucker et al., 2020). Similarly, Carroll et al. (2019) uses offline data to learn a model of the teammate's behavior and use it to train the agent to coordinate with that ally. In ad-hoc coordination, teammates' behaviors are fixed and can be estimated a priori from the dataset: the learner merely has to identify the team strategy and adopt it. Conversely, in our offline coordination setting, all the agents are learning and therefore have unknown changing behaviors: they must identify the different potential team strategies and agree on which one to follow (*Strategy Agreement*). Simultaneously, they must finetune their behaviors to one another in order to reach this team policy (*Strategy Fine-Tuning*). All this without being able to interact with other agents or the environment.

Offline MARL

Recent works have been investigating offline solutions to the MARL problem. All of these methods build on model-free single-agent approaches and constrain the policy to stay in the dataset's distribution by using either SARSA-like schemes (such as ICQ (Yang et al., 2021) and IQL (Kostrikov et al., 2021)) or policy regularization (such as CQL (Kumar et al., 2020) and TD3+BC (Fujimoto and Gu, 2021)).

Some methods investigate specific modifications to improve performance in the multi-agent setting. For instance, in the decentralized setting, MABCQ (Jiang and Lu, 2021) enforces an optimistic importance sampling modification that assumes that independent agents will strive toward similar high-rewarding states, yet since this does not discriminate between which high-rewarding state to favor, the strategy agreement issue remains. For discrete actions spaces problems, Tseng et al. (2022) propose a Transformer-based approach that learns a centralized teacher and distills its policy into independent student policies. Finally, OMAR

(Pan et al., 2022) proposes to alleviate miscoordination failure in offline MARL by adding a zeroth order optimization method on top of multi-agent CQL, achieving state-of-the-art performance on a variety of tasks. We share these works' goal of learning coordinated and efficient multi-agent teams in the offline setting. Yet, we believe that interacting learners and agents are essential to coordination and therefore take a different approach by focusing on model-based methods rather than model-free ones.

Offline model-based RL

Model-based approaches have been investigated in the single-agent offline RL setting. Notoriously, MOPO (Yu et al., 2020) proposed to learn an ensemble-based world model and use it to generate rollouts from the offline dataset to train a SAC agent (Haarnoja et al., 2018). They also proposed an uncertainty-based reward penalty to prevent the learner from exploiting the model. MOReL (Kidambi et al., 2020) takes a similar approach but prevents model abuse by learning a pessimistic MDP in which states that are outside of the dataset coverage become absorbing terminal states. COMBO (Yu et al., 2021) proposed a similar but more conservative version of MOPO by using CQL instead of SAC and learning on both generated and dataset's states. Finally, ROMI (Wang et al., 2021) also uses model-free offline RL to derive a policy from a model-based augmented offline dataset, yet they enforce additional conservatism by learning a reverse policy and dynamics model to generate rollouts that lead to target states contained in the dataset. This mitigates against generating rollouts outside of the dataset's coverage.

Our method, MOMA-PPO is in essence closest to MOPO and MOReL as it uses an online policy learning algorithm instead of an offline one. We believe that offline RL algorithms are ill-suited to learn on non-stationary data such as the one generated by updating policies be it in a world model or in a real environment. To enforce conservatism and avoid world model exploitation we use both uncertainty penalty and early rollout termination. Yet, instead of penalizing aleatoric uncertainty (the inherent uncertainty of the environment, which is nonexistent for most MuJoCo tasks with deterministic dynamics) as in MOPO, we focus on the epistemic uncertainty (due to the finite amount of training data) and estimate it by monitoring the coherence between the different models in the ensemble. Also, early rollout termination is done with timeouts rather than terminal states meaning that it does not penalize agents while still avoiding using unfeasible rollouts to train them. Finally, in contrast with the works above, MOMA-PPO is based on PPO, an on-policy method that has reliably achieved state-of-the-art performance in MARL tasks (Yu et al., 2022). It is therefore well adapted for settings in which slight changes in a teammate's policy can drastically impact the overall group behavior and quickly make previous interactions obsolete.

Model-based multi-agent RL

In the online setting, model-based approaches aim at improving sample efficiency by reducing the number of interactions with the environment. Therefore, these methods use off-policy schemes and focus on how and when to collect additional data for refining the world model and the policies (Willemsen et al., 2021; Zhang et al., 2021, 2022). In offline MARL, sample efficiency is not a consideration since the data has already been collected offline and additional data collection is not an option. Yet, we believe that model-based approaches can also be beneficial to multi-agent coordination in the offline setting as they allow multiple learners to interact through the world model.

In conclusion, this work explores coordination in offline MARL and highlights the failures of current model-free methods. Specifically, they struggle in the presence of multiple equivalent but incompatible optimal team strategies (*Strategy Agreement*), or when partial observability requires the team to adapt the behaviors demonstrated in the dataset (*Strategy Fine-Tuning*). To solve these problems, we propose MOMA-PPO which is, to our knowledge, the first model-based offline MARL approach. Our method is able to coordinate teams of offline learners and significantly outperforms model-free alternatives (even fully centralized ones). Our MOMA approach is general and some might be interested in investigating using more sample-efficient policy learning algorithms such as MOMA-SAC. Additionally, the dataset bounding box clipping and adaptive rollouts developed for MOMA-PPO might also benefit the single-agent setting. Finally, for tasks that require adapting the team's behavior, dataset coverage might be more desirable than demonstrated performance, since methods fare better on random datasets than expert ones.

5.6 Reproducibility details

The following section focuses on reproducibility and goes into detail about the implementations and experimental procedures.

5.6.1 Methods implementation

We detail here our implementation choices.

MOMA-PPO entropy bonus

For offline methods based on online RL algorithm, exploration is an important component (cf. TD3's exploration strategy) so we used an entropy bonus for PPO defined as:

```
# entropy bonus
# dimensions are batch, act_dim, n_agents.
# Instead of using closed form entropy,
# we estimate it with E(-pi log pi) were
# the expectation is sampled over pi_old
# so we have to correct with pi_new/pi_old (which is ratio!)

# we do two losses
# (clipped / not clipped like with the actor loss)
surrogate_entropy = - (ratio * new_policy).mean(0)
clipped_entropy = - (clipped_ratio * new_policy).mean(0)

entropy = torch.min(surrogate_entropy, clipped_entropy)

self.entropy_alpha += self.ppo_entropy_bonus_coeff*(self.
                                    ppo_entropy_target - entropy).detach
                                    ()
self.entropy_alpha.data.clamp_min_(0.)

# minus sign because we minimize the expressions
# i.e. max(ent) = min(-ent)
entropy_bonus = - entropy * self.entropy_alpha
```

with an entropy bonus coefficient of 0.001 and an entropy target of -6 and -4 for respectively Ant and Reacher tasks.

MOMA-PPO action penalty

Since entropy computation can become numerically unstable for squashed actions close to the Tanh bounds, PPO uses action penalty instead of Tanh squashing to keep actions close to the -1, 1 range (with an action penalty coefficient of 1.):

```
delta = (1. - actions.abs())
action_bound_error = ((delta < 0.).to(torch.float32) * delta**2).sum(1,
                                        keepdim=True)

surrogate_action_bound_error = (ratio * action_bound_error).mean(0)
clipped_action_bound_error = (clipped_ratio * action_bound_error).mean(0)

action_bound_error = torch.max(surrogate_action_bound_error,
                                        clipped_action_bound_error)
action_penalty = self.ppo_action_penalty_coeff * action_bound_error
```

General Advantage Estimation

We show below how we modified the general advantage estimation to account for rollout termination (indicated by `time_out_masks`).

```
# initial (end) running returns is the next state value
running_returns = next_values[-1] * masks[-1]

# initial (end) advantage is 0 because no difference in value and return
running_advants = 0

for t in reversed(range(0, len(rewards))):

    # We are going reverse so if done, only reward because end of
    # episode if timeout, we stop accumulation and use value as
    # bootstrap like in initialization
    running_returns = rewards[t] + self.discount * masks[t] * (
                                        running_returns * time_out_masks[
                                        t] + (1. - time_out_masks[t]) *
                                        next_values[t] * masks[t])

    returns[t] = running_returns

    ## No accumulation here and timeout doesn't influence next_state value
    running_delta = rewards[t] + (self.discount * next_values[t] * masks[t
                                        ]) - values[t]

    ## if timeout running_advants goes back to running_delta because
    # we do not have extra rewards to estimate it
    # (cf initialization above)
    running_advants = running_delta + (self.discount * self.lamda *
                                        running_advants * masks[t]) *
                                        time_out_masks[t]

    advants[t] = running_advants
```

Memory module for partial observability

To handle partial observability we use observation-action histories h_t^i of size ten (i.e. the ten past observation-action pairs). These histories are processed with self-attention followed by soft-attention to yield embeddings of size $e_h = 128$ that are concatenated to the current state before being fed to the policy and value networks. We use a first linear layer to encode h_t^i to $\mathbb{R}^{e_h}$ and add positional encodings (Vaswani et al., 2017). Query, Key, and Value networks are linear layers and we follow Vaswani et al. (2017)'s Scaled Dot-Product Attention with skip connection and layer-norm. Finally, the resulting self-attentions are aggregated using soft-attention with the soft-key network being a bias-less linear layer and the soft-queries are e_h normally initialized trainable parameters.

Note that policy and value networks use (and backprop through) the same memory module but target networks have their own target memory module (that tracks the memory module with Polyak updates just like regular target networks). The memory learning rate is $1e^{-4}$ for all the algorithms.

Finetuning MAIQL: MAIQL-ft

We follow Kostrikov et al. (2021)'s finetuning procedure and use the ground truth simulator to generate the rollout. However, the rollouts are still generated using MOMA's Dyna-like approach described in Section 5.2 rather than generating length 1000 rollouts from the initial state distribution. Indeed, we consider the model-based offline setting and not the offline-to-online finetuning setting. Therefore it is unfeasible to assume that the task's ground-truth initial state distribution is known or that it is possible to learn a world model that remains accurate over 1000 simulation steps.

Opensource baselines

For the baseline implementations, we followed the official repositories at:

- `https://github.com/sfujim/TD3_BC`,
- `https://github.com/ling-pan/OMAR/`,
- and `https://github.com/ikostrikov/implicit_q_learning`.

5.6.2 Hyperparameters, Tuning, and Training

We detail here the training procedures.

Hyperparameters and tuning

Unless specified otherwise, networks are two-layers 256 units ReLu MLPs. We use Adam optimizer (Kingma and Ba, 2014) with default hyperparameters (except learning rates) and

a batch size of 256. We clip gradient norms to 1. We use a discount factor of 0.99 and a target update coefficient of 0.005 for the Polyak averaging.

World model learning. World models use four layers and 1024 units. World models use a learning rate of $3e^{-5}$ and are trained for $3e^{6}$ steps.

Policy learning. MAPPO learning rate was finetuned on MAMuJoCo online task `halfcheetah-v2_2x3_full` with a grid search across $[1e^{-6}, 5e^{-6}, 1e^{-5}, 5e^{-5}, 1e^{-4}, 5e^{-4}, 1e^{-3}]$. We kept the value of $5e^{-5}$ for MOMA-PPO for all experiments. PPO uses rollouts length of 1000, 5 epochs per update, 2000 transitions between updates, a clip value of 0.2, a λ value of 0.98, and a critic loss coefficient of 0.5.

MAIQL showed quite unstable and we had to finetune its learning rate extensively depending on the tasks. We tried the range $[1e^{-4}, 3e^{-4}, 1e^{-5}, 5e^{-5}, 1e^{-6}]$ on ant-expert for MAIQL and MAIQL-ft and kept $3e^{-4}$. For the results on ant-expert with full observability, we had to discard one collapsed unstable seed for MAIQL and retrain another seed. For reacher tasks we tried $[5e^{-5}, 1e^{-4}, 3e^{-4}, 5e^{-4}, 1e^{-3}]$ and kept $3e^{-4}$. IQL and MAIQL use an expectile value of 0.7 and an AWR temperature of 3.

All other methods use their default learning rate of $3e^{-4}$. ITD3+BC uses a BC regularization parameter of 2.5, a policy frequency update of 2, a policy noise of 0.2, and a noise clip value of 0.5. ICQL uses a coefficient of 1 (and so does IOMAR's CQL component) for all tasks except for ant-expert tasks where we had to tune it between $[0.1, 0.5, 1, 5]$ and kept a value of 5 (same for IOMAR). Additionally, CQL uses an LSE temperature of 1, 10 sampled actions, and a sample noise level of 0.2. Finally, IOMAR uses a coefficient of 1, 2 iterations, a μ value of 0, a σ value of 2, 20 samples, and 5 elites.

TD3 (and thus CQL and OMAR) policies are deterministic with Tanh squashing while IQL and PPO use Gaussians with state-dependent variance. IQL uses Tanh squashing while PPO does not (see below).

Model-free methods are trained for $1e^{6}$ learning steps while MOMA-PPO is trained for $1e^{5}$ learning steps which correspond to roughly $2e^{8}$ collected world model interactions.

Compute

Our longest MOMA-PPO training took 6 days on a Tesla P100-PCIE-12GB GPU. For comparison our longest IOMAR run took 38 hours on a Tesla V100-SXM2-32GB GPU and our longest MAIQL run took 40 hours on a Tesla V100-SXM2-32GB GPU and five days on a Tesla P100-PCIE-12GB GPU for MAIQL-ft. Training a world model took at most three days on a Tesla P100-PCIE-12GB GPU. Note that training times are also impacted by how often we estimate performance for the learning curves and we do it much more often for MOMA-PPO (5 episodes every 50 learning steps) than for other methods (10 episodes every 5000 learning steps).

5.7 Raw results

This section focuses on transparency and provides the raw results of the experiments.

5.7.1 Learning Curves

Figues 5.5 and 5.7 show the learning curves for the Reacher and Ant environments respectively. We use 5 episodes to evaluate MOMA-PPO every 50 learning steps and 10 episodes every 5000 learning steps to evaluate the other methods (this is why MOMA-PPO curves might look noisier). We used a smoothing factor of .8 for all curves. We report the mean over three seeds and the shaded area represents ± the standard error of the mean. We train MOMA-PPO for $1e^5$ training steps because it is on-policy while the off-policy methods are trained for $1e^6$ training steps. MAIQL-ft training is $2e^6$ steps (half offline and half online fine-tuning).

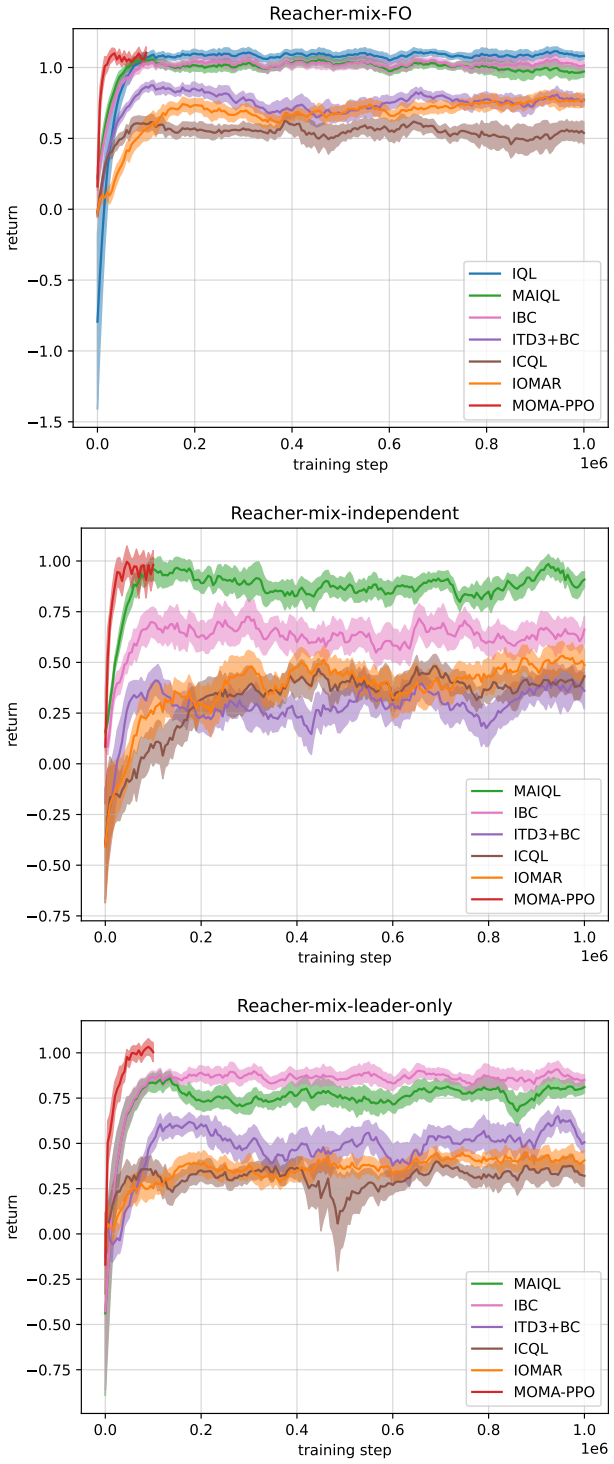


Figure 5.5: Learning Curves for two-agent Reacher. Mean and standard error of the mean on three seeds.

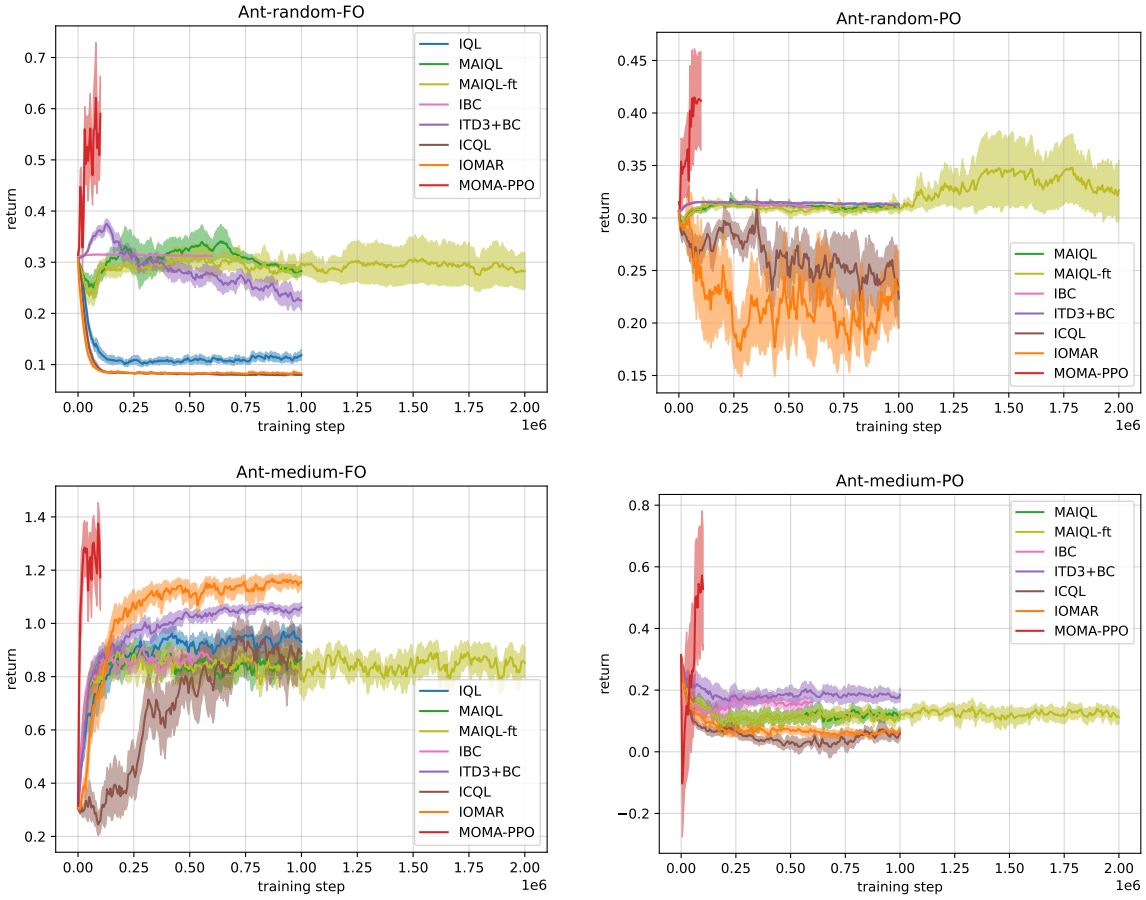


Figure 5.6: Learning curves for four-agent Ant on random and medium datasets. Mean and standard error of the mean on three seeds.

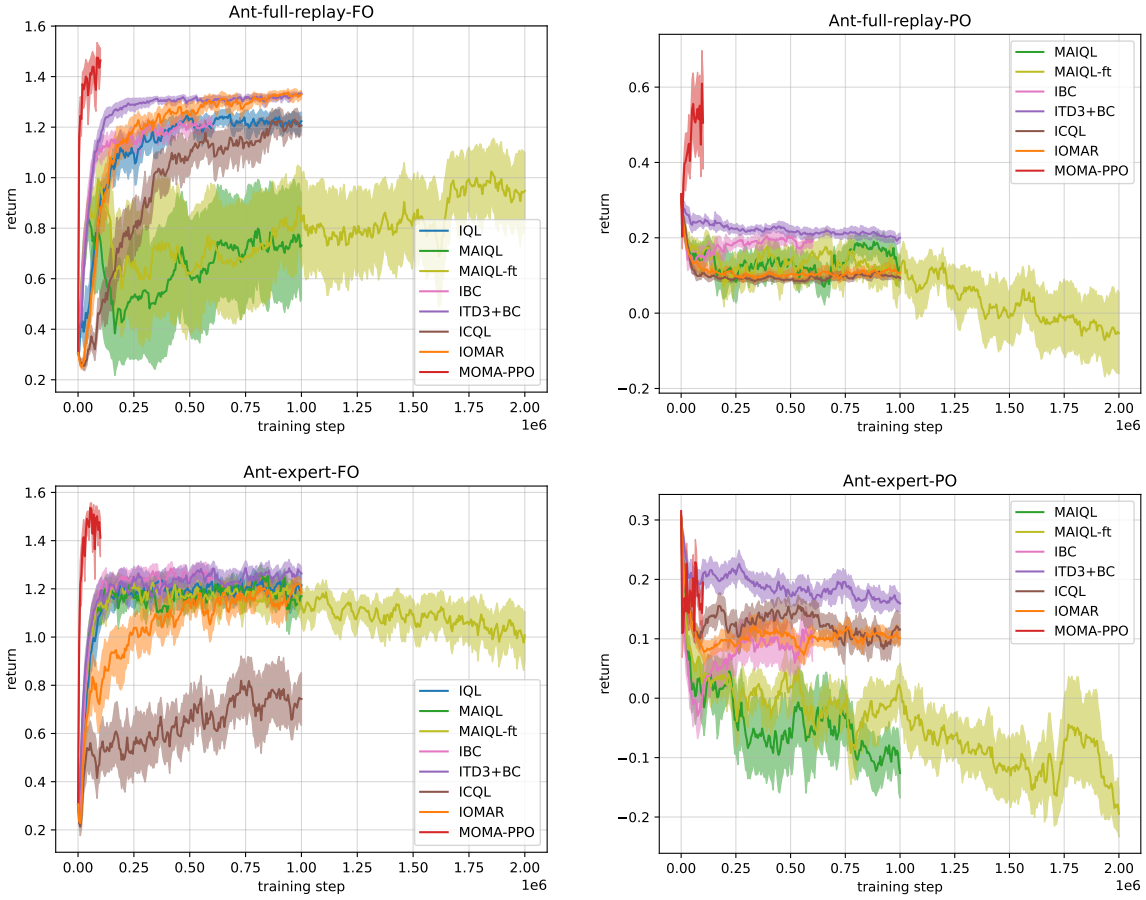


Figure 5.7: Learning curves for four-agent Ant on full replay and expert datasets. Mean and standard error of the mean on three seeds.

5.7.2 Ablations

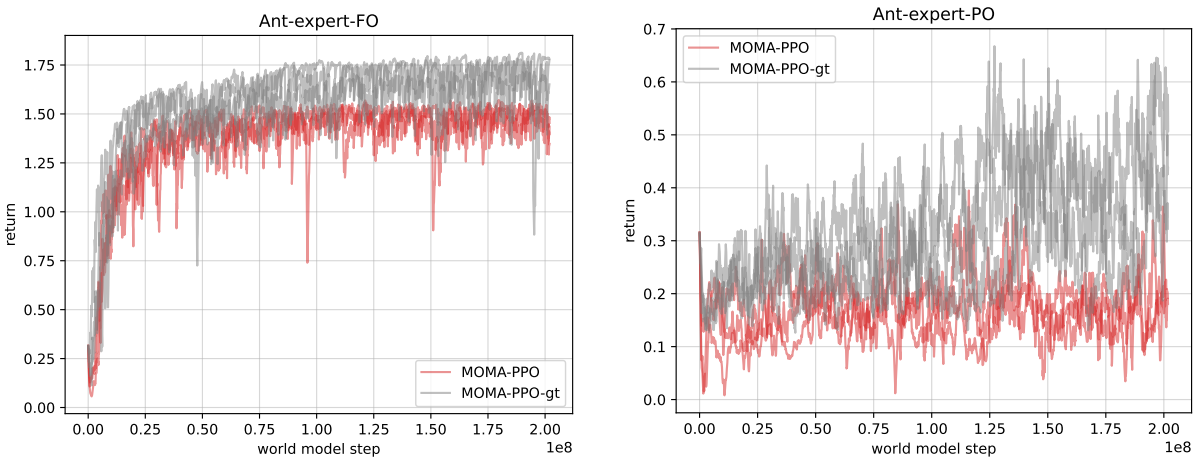


Figure 5.8: Impact of using the ground truth world model vs. the learned world model. "gt" stands for "ground truth" and means that the corresponding runs use the ground truth world model. The learning curves are with respect to generated transitions (either from the ground truth or the learned world model). Here we show each individual run instead of the usual mean and standard error of the mean.

		MOMA-PPO	MOMA-PPO-gt
(FO)	ant-random	0.52 ± 0.07	**1.17 ± 0.03**
	ant-medium	1.29 ± 0.06	**1.68 ± 0.03**
	ant-full-replay	1.42 ± 0.07	**1.66 ± 0.03**
	ant-expert	1.49 ± 0.01	**1.71 ± 0.04**
(PO)	ant-random	0.42 ± 0.05	0.47 ± 0.01
	ant-medium	0.54 ± 0.19	0.81 ± 0.20
	ant-full-replay	0.46 ± 0.10	**0.84 ± 0.07**
	ant-expert	0.18 ± 0.00	**0.43 ± 0.06**

Table 5.6: Mean scores and standard error of the mean at the end of training with and without the use of the ground truth simulator. Evaluations are over 100 episodes.

Figure 5.8 and Table 5.6 compare using a learned world model with having access to the ground truth simulator to generate the rollouts. It appears that MOMA-PPO performance can be further improved provided that we learn better world models.

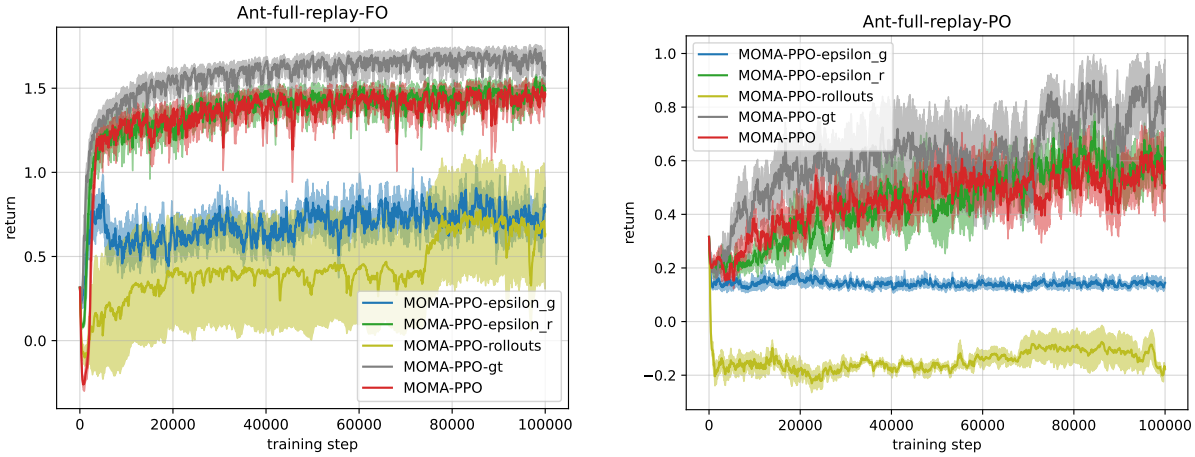


Figure 5.9: Impact of the different uncertainty-based techniques to prevent model exploitation by the learning algorithm. "gt" stands for "ground truth" and means that the corresponding runs use the ground truth world model. "epsilon_g" and "epsilon_r" respectively mean that we set $\lambda_g = 0$ and $\lambda_r = 0$. "rollouts" means that we removed the adaptive rollouts component (i.e. $l_\epsilon = \infty$). Mean and standard error of the mean on three seeds.

Figure 5.9 shows ablations on MOMA-PPO. It appears that λ_r penalty can be removed without hurting performance, indeed it is already encapsulated in λ_g. The other components of MOMA-PPO such as λ_g uncertainty penalty on the reward, or the adaptive rollouts that terminate if the uncertainty crosses a threshold, are mandatory to reach satisfactory performance.

Note that running experiments without clipping the world model predictions to the datasets' bounding box was impossible as the magnitude of the predicted state exploded after a few rollout steps. This could be mitigated by reducing the exploration of the PPO policies so that the agents stay closer to the dataset distribution where the world model does not predict such extreme states.

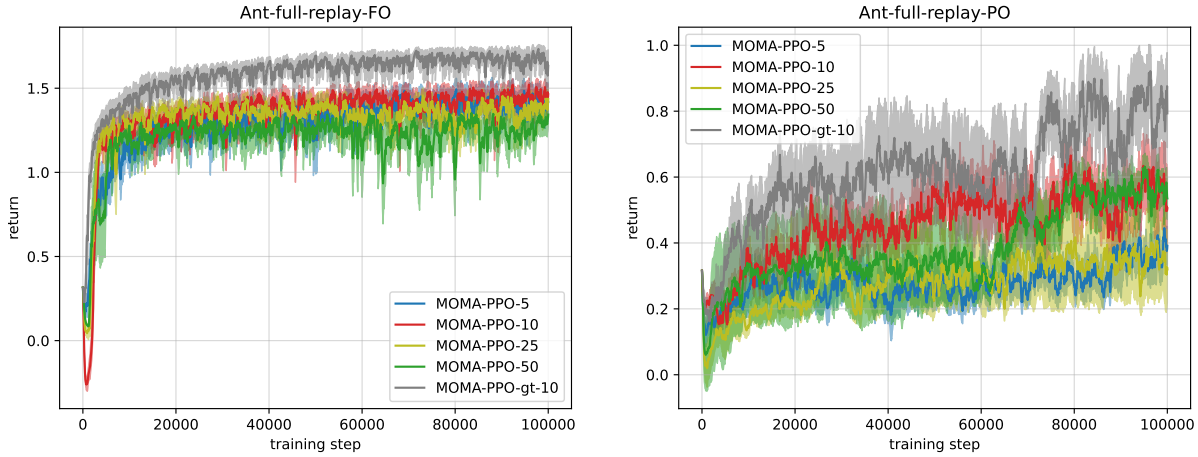


Figure 5.10: Impact of different maximum rollout lengths when generating the synthetic interactions. MOMA-PPO-x indicates a maximum rollout length of x. "gt" stands for "ground truth" and means that the corresponding runs use the ground truth world model. Mean and standard error of the mean on three seeds.

Figure 5.10 shows MOMA-PPO trainings for different maximum rollout lengths. It appears that varying the maximum length from 5 to 50 does not significatively impact MOMA-PPO performance on the tasks we investigated.

Chapter 6

Coordinating in the absence of rewards or demonstrations

This chapter features the work that we published in Barde et al. (2022). Here we investigate how agents can coordinate in the absence of explicit rewards or demonstrations. Specifically, we investigate the emergence of communication between an agent that receives rewards but cannot directly affect the environment to maximize it, and an agent that can act in the environment but receives no rewards. While agents must collaborate and coordinate if they are to solve the task, they have no shared incentives – one of them has no reward nor explicit learning signal – nor the possibility to demonstrate and rely on observational reinforcement. Learning in this setting seems unfeasible, yet, we show that it is possible by leveraging intrinsic shared intent priors and structuring the agents interactions.

We are interested in interactive agents that learn to coordinate, namely, a *builder* – which performs actions but ignores the goal of the task, i.e., has no access to rewards – and an *architect* which guides the builder towards the goal of the task. We define and explore a formal setting where artificial agents are equipped with mechanisms that allow them to simultaneously learn a task while at the same time evolving a shared communication protocol. Ideally, such learning should only rely on high-level communication priors and be able to handle a large variety of tasks and meanings while deriving communication protocols that can be reused across tasks.

Humans are notoriously successful at teaching – and learning from – each other. This enables skills and knowledge to be shared and passed along through generations, being progressively refined toward mankind's current state of proficiency. People can teach and be taught in situations where there is no shared language and very little common ground, such as a parent teaching a baby how to stack blocks during play. Experimental Semiotics, a line of work that studies the forms of communication that people develop when they cannot use pre-established ones, reveals that humans can even teach and learn without direct reinforce-

ment signals, demonstrations, or a shared communication protocol (Galantucci and Garrod, 2011). Vollmer et al. (2014) for instance investigate a Co-Construction (CoCo) game experiment where an architect must rely only on arbitrary instructions to guide a builder toward constructing a structure. In this experiment, both the task of building the structure and the meanings of the instructions – with which the architect guides the builder – are simultaneously learned throughout interactions. Such flexible teaching – and learning – capabilities are essential to autonomous artificial agents if they are to master an increasing number of skills without extensive human supervision. As a first step toward this research direction, we draw inspiration from the CoCo game and propose the Architect-Builder Problem (ABP): an interactive learning setting that models agents' interactions with MDPs (Puterman, 2014). In the ABP, learning has to occur in a social context through observations and communication, in the absence of direct imitation or reinforcement (Bandura and Walters, 1977). Specifically, the constraints of the ABP are: (1) the builder has absolutely no knowledge about the task at hand (no reward and no prior on the set of possible tasks), (2) the architect can only interact with the builder through communication signals (cannot interact with the environment or provide demonstrations), and (3) the communication signals have no pre-defined meanings (nor belong to a set of known possible meanings). (1) sets this work apart from Reinforcement Learning and even MARL in which explicit rewards are available to all the agents. (2) implies the absence of teleoperation or third-person demonstrations and thus distinguishes the ABP from Imitation Learning and Inverse Reinforcement Learning. Finally, (3) prevents the architect from relying on a fixed communication protocol since the meanings of instructions must be negotiated.

These constraints make ABP an appealing setting to investigate Human-Robot Interaction (HRI) problems where "a learner tries to figure out what a teacher wants them to do" (Goodrich and Schultz, 2008; Grizou et al., 2013; Cederborg and Oudeyer, 2014). In particular, the challenge of Brain-Computer Interfaces (BCIs), where users use brain signals to control virtual or robotic agents in sequential tasks is well captured by the ABP (Katyal et al., 2014; deBettencourt et al., 2015; Mishra and Gazzaley, 2015; Muñoz-Moldes and Cleeremans, 2020; Chiang et al., 2021). In BCIs, (3) is referred to as the "calibration problem" and is usually tackled with supervised learning to learn a mapping between signals and meanings. As this calibration phase is often laborious and impractical for users, current approaches investigate calibration-free solutions where the mapping is learned interactively (Grizou et al., 2014a; Xie et al., 2021). Yet, these works consider that the user (i.e., the architect) is fixed, in the sense that it does not adapt to the agent (i.e., the builder) and uses a set of pre-defined instruction – and/or feedback – meanings that the agent must learn to map to signals. Contrarily, in our ABP formulation, the architect is dynamic and, as interactions unfold, must learn to best guide a learning builder by tuning the meanings of instructions according to the builder's reactions. In that sense, ABP provides a more complete computational model of agent-agent or human-agent interaction.

With all these constraints in mind, we propose Architect-Builder Iterated Guiding (ABIG),

an algorithmic solution to ABP when both agents are AIs. ABIG is inspired by the field of experimental semiotics and relies on two high-level coordination priors: *shared intent* and *interaction frames*. Shared intent refers to the fact that, although the builder ignores the objective of the task to fulfill, it will assume that its objective is aligned with the one of the architects. This assumption is characteristic of cooperative tasks that are shown to be a necessary condition for the emergence of communication both in practice (Foerster et al., 2016; Cao et al., 2018) and in theory (Crawford and Sobel, 1982). Specifically, the builder should assume that the architect is guiding it toward a shared objective. Knowing this, the builder must reinforce the behavior it displays when guided by the architect. We show that the builder can efficiently implement this by using imitation learning on its own guided behavior. Because the builder imitates itself, we call it self-imitation. The notion of *interaction frames* (also called *pragmatic frames*) states that agents that interact in sequence can more easily interpret the interaction history (Bruner, 1985; Vollmer et al., 2016). In ABIG, we consider two distinct stationary interaction frames. Here stationary refers to the fact that, when one agent learns, the other agent's behavior is fixed. During the first frame (the modeling frame), the builder is fixed and the architect learns a model of the builder's message-conditioned behavior. During the second frame (the guiding frame), the architect is fixed and the builder learns to be guided via self-imitation learning.

We show that ABIG results in a low-level, high-frequency, guiding communication protocol that not only enables an architect-builder pair to solve the task at hand and can also be used to solve unseen tasks. Our contributions are:

- The formulation of the Architect-Builder Problem, an interactive learning setting to study the mechanisms by which artificial agents can coordinate and learn to solve a task while simultaneously deriving a communication protocol.
- Architect-Builder Iterated Guiding, an algorithmic solution to the ABP.
- An analysis of ABIG's key learning mechanisms.
- An evaluation of ABIG on a construction environment where we show that ABIG agents evolve communication protocols that generalize to unseen harder tasks.
- A detailed analysis of ABIG's learning dynamics and impact on the mutual information between messages and actions.

6.1 The Architect-Builder Problem

In this section, we present our novel interactive learning setting.

6.1.1 The formalism

We consider a multi-agent setup composed of two agents: an architect and a builder. Both agents observe the environment state s but only the architect knows the goal at hand. The

architect cannot take action in the environment but receives the environmental reward r whereas the builder does not receive any reward and has therefore no knowledge about the task at hand. In this asymmetrical setup, the architect can only interact with the builder through a communication signal m sampled from its policy $\pi_{A}(m|s)$. These messages, which have no a priori meanings, are received by the builder which acts according to its policy $\pi_{B}(a|s,m)$. This makes the environment transition to a new state s' sampled from $P_{E}(s'|s,a)$, and the architect receives reward r. Messages are sent at every time step. The ABP setting, as well as the CoCo game that inspired it, are sketched in Figure 6.1. The differences between the ABP setting and the commons MARL and IRL settings are also highlighted in Figure 6.1.

6.1.2 The BuildWorld

We conduct our experiments in BuildWorld a 2D construction grid-world of size $(w \times h)$. At the beginning of an episode, the agent and N_b blocks are spawned at different random locations. The agent can navigate in this world and grasp blocks by activating its gripper while on a block. More specifically, the agent has a discrete action space $\mathcal{A}$ of size 6: the first four actions control the direction of navigation (North, South, East, West); the fifth action toggles the gripper (grasp/drop) and the last one is a "do nothing" action. At each time step, the agent observes its position in the grid, its gripper state as well as the position of all the blocks and if they are grasped ($|\mathcal{S}| = 3 + 3N_b$).

6.1.3 The tasks

BuildWorld contains four different *training tasks*:

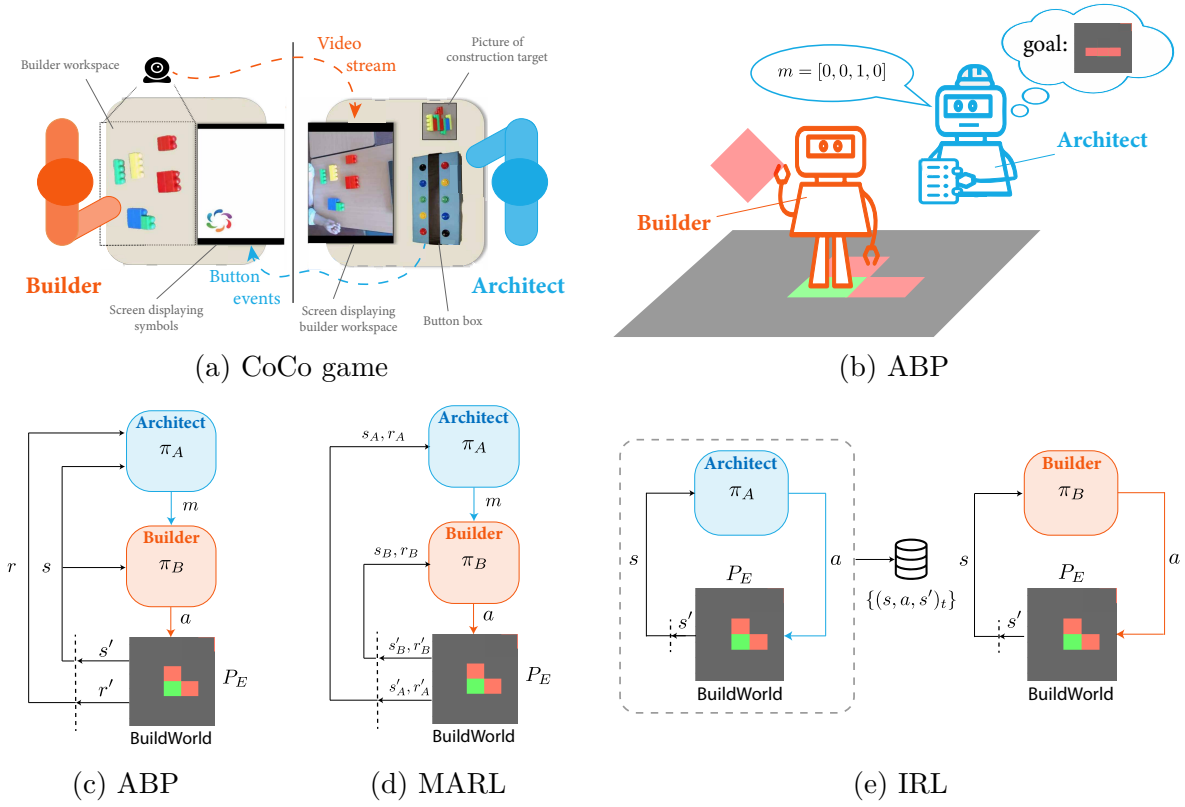
1. 'Grasp': The agent must grasp any of the blocks;
2. 'Place': The agent must place any block at a specified location in the grid;
3. 'H-Line': The agent must place all the blocks in a horizontal line configuration;
4. 'V-line': The agent must place all the blocks in a vertical line configuration.

BuildWorld also has a harder fifth *testing task*:

5. '6-blocks-shapes', which consists of more complex configurations and that is used to challenge an algorithm's transfer abilities.

For all tasks, rewards are sparse and only given when the task is completed.

This environment encapsulates the interactive learning challenge of ABP while removing the need for complex perception or locomotion. In the RL setting, where the same agent acts and receives rewards, this environment would not be very impressive. However, it remains to be shown that the tasks can be solved in the challenging learning setting of ABP (with a reward-less builder and an action-less architect).



(a) CoCo game (b) ABP

(c) ABP (d) MARL (e) IRL

Figure 6.1: (a) **Schematic view of the CoCo Game.** The architect and the builder should collaborate in order to build the construction target while located in different rooms. The architecture has a picture of the target structure while the builder has access to the blocks. The architect monitors the builder's workspace via a camera (video stream) and can communicate with the builder only through the use of 10 symbols (button events). (b) **Schematic view of the ABP.** The architect must learn how to use messages to guide the builder while the builder needs to learn to make sense of the messages in order to be guided by the architect. (c) **Interaction diagram in the ABP.** The architect communicates messages (m) to the builder. Only the builder can act (a) in the environment. The builder conditions its action on the message sent by the builder ($\pi_B(a|s, m)$). The builder never perceives any reward from the environment and only the architect perceives the reward signal (r). (d) **MARL framework.** Both the architect and the builder have access to environmental rewards r_A and r_B. Which would contradict the fact that the builder ignores everything about the task at hand; (e) **IRL framework.** The architect needs to provide demonstrations. The architect does not exchange messages with the builder. The builder uses the demonstrations $\{(s, a, s')_t\}$ to learn the desired behavior.

6.1.4 The communication

The architect guides the builder by sending messages m which are one-hot vectors of size $|\mathcal{V}|$ ranging from 2 to 72, see Subsection 6.4.1 for the impact of this parameter.

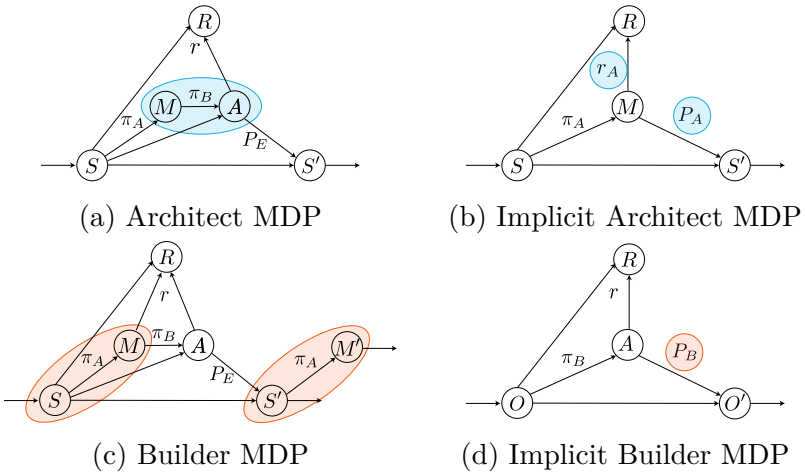
6.1.5 The additional assumptions

In order to focus on the architect-builder interactions and the learning of a shared communication protocol, the architect has access to $P_{_E}(s'|s,a)$ and to the reward function $r(s,a)$ of the goal at hand. This assumes that, if the architect were to act in the environment instead of the builder, it would be able to quickly figure out how to solve the task from model-based planning. This assumption is compatible with the CoCo game experiment (Vollmer et al., 2014) where human participants, and in particular the architects, are known to have such world models.

6.2 Architect-Builder Iterated Guiding

In this section we present ABIG, our solution to the Architect-Builder Problem.

6.2.1 Analytical description



(a) Architect MDP (b) Implicit Architect MDP

(c) Builder MDP (d) Implicit Builder MDP

Figure 6.2: Agent's Markov Decision Processes. Highlighted regions refer to MDP coupling. (a) The architect's transitions and rewards are conditioned by the builder's policy $\pi_{_B}$. (b) Architect's MDP where transition and reward models implicitly account for the builder's behavior. (c-d) The builder's transition model depends on the architect's message policy $\pi_{_A}$. The builder's learning signal r is *unknown*.

Agents' MDPs. In the Architect-Builder Problem, agents are operating in different, yet coupled, MDPs. These MDPs depend on their respective point of view (see Figure 6.2). From the point of view of the architect, messages are actions that influence the next state as well as the reward (see Figure 6.2 (a)). The architect knows the environment transition function $P_{_E}(s'|s,a)$, and $r(s,a)$, the true reward function associated with the task that does not depend explicitly on messages. It must thus additionally reason about the effect of its messages on the builder's actions since these actually drive the reward and the next states (see Figure 6.2 (b)). On the other hand, the builder's state is composed of the environment state and the message, which makes estimating state transitions challenging as one must also capture the message dynamics (see Figure 6.2 (c)). Yet, the builder can assume that the architect picks messages based on the current environment state. The equivalent transition and reward models, when available, are given below:

$$\left.\begin{aligned} P_{_A}(s'|s,m) &= \sum_{a\in\mathcal{A}} \tilde{\pi}_{_B}(a|s,m) P_{_E}(s'|a,s) \\ r_{_A}(s,m) &= \sum_{a\in\mathcal{A}} \tilde{\pi}_{_B}(a|s,m) r(s,a) \end{aligned}\right\} \quad \text{with} \quad \tilde{\pi}_{_B}(a|s,m) \triangleq P(a|s,m) \tag{6.1}$$

$$P_{_B}(s',m'|s,m,a) = \tilde{\pi}_{_A}(m'|s') P_{_E}(s'|s,a) \quad \text{with} \quad \tilde{\pi}_{_A}(m'|s') \triangleq P(m'|s') \tag{6.2}$$

where subscripts A and B respectively refer to the architect and the builder. $\tilde{x}$ denotes that x is unknown and must be approximated. From the builder's point of view, the reward – denoted $\tilde{r}$ – is unknown. This prevents the use of classical RL algorithms.

To derive these results we use the laws of total probabilities and conditional probabilities:

$$\begin{aligned} P_{_A}(s'|s,m) &= \sum_{a\in\mathcal{A}} P(s',a|s,m) \\ &= \sum_{a\in\mathcal{A}} P(s'|a,s,m) P(a|s,m) \\ &= \sum_{a\in\mathcal{A}} P_{_E}(s'|a,s) \tilde{\pi}_{_B}(a|s,m) \end{aligned}$$

Where the final equality uses the knowledge that next-states only depend on states and the

builder's actions.

$$\begin{aligned} r_{_A}(s,m,s') &\triangleq \mathbb{E}[R|s,m,s'] \\ &= \int_{\mathbb{R}} rP(r|s,m,s')dr \\ &= \int_{\mathbb{R}} r \sum_{a\in\mathcal{A}} P(r,a|s,m,s')dr \\ &= \int_{\mathbb{R}} r \sum_{a\in\mathcal{A}} P(r|s,m,a,s')P(a|s,m,s')dr \\ &= \int_{\mathbb{R}} r \sum_{a\in\mathcal{A}} P(r|s,a,s')\tilde{\pi}_{_B}(a|s,m)dr \\ &= \sum_{a\in\mathcal{A}} \tilde{\pi}_{_B}(a|s,m) \int_{\mathbb{R}} rP(r|s,a,s')dr \\ &= \sum_{a\in\mathcal{A}} \tilde{\pi}_{_B}(a|s,m)r(s,a,s') \end{aligned}$$

$$\begin{aligned} P(s',m'|s,m,a) &= P(m'|s',s,m,a)P(s'|s,m,a) \\ &= P(m'|s')P(s'|s,a) \\ &= \tilde{\pi}_{_A}(m'|s')P_{_E}(s'|s,a) \end{aligned}$$

Shared Intent and Interaction Frames. It follows from Eq. (6.1) that, provided that it can approximate the builder's behavior, the architect can compute the reward and transition models of its MDP. It can then use these to derive an optimal message policy $\pi^*_{_A}$ that would maximize its objective:

$$\pi^*_{_A} = \arg\max_{\pi_{_A}} G_{_A} = \arg\max_{\pi_{_A}} \mathbb{E}[\sum_t \gamma^t r_{_A,t}] \tag{6.3}$$

$\gamma \in [0,1]$ is a discount factor and the expectation can be thought of in terms of $\pi_{_A}$, $P_{_A}$, and the initial state distribution. However, the expectation can also be thought of in terms of the corresponding trajectories $\tau \triangleq \{(s,m,a,r)_t\}$ generated by the architect-builder interactions. In other words, when using $\pi^*_{_A}$ to guide the builder, the architect-builder pair generates trajectories that maximize $G_{_A}$.

The builder has no reward signal to maximize, yet, it relies on a shared intent prior and should assume that its objective is the same as the one of the architect:

$$G_{_B} = G_{_A} = \mathbb{E}_\tau[\sum_t \gamma^t r_{_A,t}] = \mathbb{E}_\tau[\sum_t \gamma^t \tilde{r}_t] \tag{6.4}$$

where the expectations are taken with respect to trajectories τ of architect-builder interactions. Therefore, under the shared intent prior, architect-builder interactions where the

architect uses π_A^* to maximize G_A, also maximize G_B. This means that the builder can interpret these interaction trajectories as demonstrations that maximize its unknown reward function $\tilde{r}$. Consequently, the builder can reinforce the desired behavior – towards which the architect guides it – by performing self-Imitation Learning[1] on the interaction trajectories τ.

Note that in Eq. (6.1), the architect's models can be interpreted as expectations with respect to the builder's behavior. Similarly, the builder's objective depends on the architect's guiding behavior. This makes one agent's MDP highly non-stationary and the agent must adapt its behavior if the other agent's policy changes. To palliate this, agents rely on interaction frames which means that, when one agent learns, the other agent's policy is fixed to restore stationarity. The equivalent MDPs for the architect and the builder are respectively $\mathcal{M}_A = \langle \mathcal{S}, \mathcal{V}, P_A, r_A, \gamma \rangle$ and $\mathcal{M}_B = \langle \mathcal{S} \times \mathcal{V}, \mathcal{A}, P_B, \emptyset, \gamma \rangle$. With $\pi_A : \mathcal{S} \to \mathcal{V}$, $P_A : \mathcal{S} \times \mathcal{V} \to [0, 1]$, $r_A : \mathcal{S} \times \mathcal{V} \to [0, 1]$, $\pi_B : \mathcal{S} \times \mathcal{V} \to \mathcal{A}$ and $P_B : \mathcal{S} \times \mathcal{V} \times \mathcal{A} \to [0, 1]$ where $\mathcal{S}, \mathcal{A}$ and $\mathcal{V}$ are respectively the sets of states, actions, and messages.

6.2.2 The algorithm

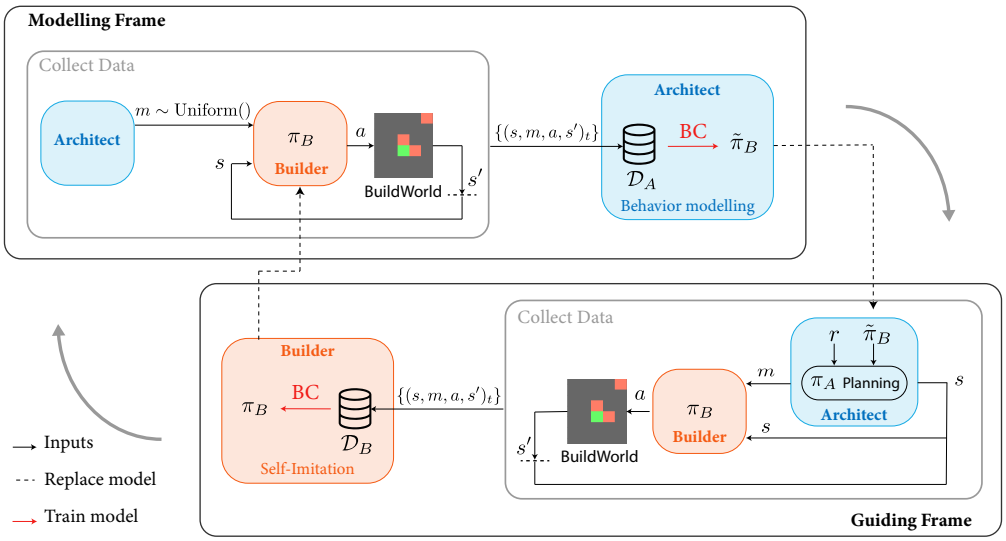


Figure 6.3: Architect-Builder Iterated Guiding. Agents iteratively interact through the modeling and guiding frames. In each frame, one agent collects data and improves its policy while the other agent's behavior is fixed.

ABIG iteratively structures the interactions between a builder-architect pair into interaction frames. Each iteration starts with a *modeling frame* during which the architect learns a

[1] not to be confused with Oh et al. (2018) which is an off-policy actor-critic algorithm promoting exploration in single-agent RL.

model of the builder. Directly after, during the *guiding frame*, the architect leverages this model to produce messages that guide the builder. On its side, the builder stores the guiding interactions to train and refine its policy π_{B}. The interaction frames are described below and the algorithm is illustrated in Figure 6.3. The pseudo-code is reported in Algorithm 4.

Modeling Frame. The architect records a dataset of interactions $\mathcal{D}_{A} \triangleq \{(s, m, a, s')_t\}$ by sending random messages m to the builder and observing its reaction. After collecting enough interactions, the architect learns a model of the builder $\tilde{\pi}_{B}$ using Behavioral Cloning.

Guiding Frame. During the guiding frame, the architect observes the environment states s and produces messages so as to maximize its return (see Eq. (6.3)). The policy of the architect is a MCTS algorithm (Kocsis and Szepesvári, 2006) that searches for the best message by simulating the reaction of the builder using $\tilde{a} \sim \tilde{\pi}_{B}(\cdot|m, s)$ alongside the dynamics and reward models. During this frame, the builder stores the interactions in a buffer $\mathcal{D}_{B} \triangleq \{(s, m, a, s')_t\}$. At the end of the guiding frame, the builder self-imitates by updating its policy π_{B} with BC on $\mathcal{D}_{B}$.

The resulting method (ABIG) is general and can handle a variety of tasks while not restricting the kind of communication protocol that can emerge. Indeed, it only relies on a few high-level priors, namely, the architect's access to environment models, shared intent, and interaction frames.

Ablations. In addition to ABIG we also investigate two control settings: ABIG *-no-intent* – the builder interacts with an architect that disregards the goal and therefore sends random messages during training. At evaluation time, the architect has access to the exact model of the builder ($\tilde{\pi}_{B} = \pi_{B}$) and leverages it to guide it towards the evaluation goal (the architect no longer disregards the goal). And *random* – the builder takes random actions. The comparison between ABIG and ABIG-no-intent measures the impact of doing self-imitation on guiding versus on non-guiding trajectories. The random baseline is used to provide a performance lower bound that indicates the task's difficulty.

6.2.3 Practical considerations

Behavioral Cloning. BC minimizes the cross-entropy loss with Adam optimizer (Kingma and Ba, 2014). Specifically, the dataset is split into training (70%) and validation (30%) sets, and training is stopped if the validation accuracy has not improved after a 'wait for' number of epochs. For a training dataset $\mathcal{D}$ of size N the policy model $\tilde{\pi}_{\theta}$ parametrized by θ minimizes the BC loss given by:

$$J(\theta) = \frac{1}{N} \sum_{\mathcal{D}} -\log \tilde{\pi}_{\theta}(a|s, m) \tag{6.5}$$

Networks are re-initialized before each BC training.

Monte Carlo Tree Search. The architect's MCTS uses UCT and relies on heuristics rather than Monte-Carlo rollouts to estimate the value of states. Specifically, nodes are labeled by the environment's states and they are expanded by selecting messages. Selecting message m from a node with label s yields a builder action according to the architect's builder model $a \sim \tilde{\pi}_{B}(a|s,m)$, this sampled action in turn yields the label of the child node according to the environment's transition model $s' \sim P_{E}(s'|s,a)$. We repeat this process until we select a message that was never selected from the current node or we sample a next state that does not correspond to a child node yet. In both of these cases, a new node has to be created. We estimate the value of the new node using an engineered heuristic that estimates the return of an optimal policy $\pi^{*}(a|s)$ from state s. This value is scaled down by a factor of two to avoid overestimation: the builder's policy may not allow the architect to have it follow π^{*}. This estimated value for a newly created node at depth l is back-propagated as a return to parents node at depth k according to:

$$G^{k} = \sum_{\tau=0}^{l-1-k} \gamma^{\tau} r_{k+1+\tau} + \gamma^{l-k} v^{l} \qquad k = l, ..., 0 \tag{6.6}$$

where r_j is the reward collected from node at depth j to child node at depth $j+1$. From a node with label s we select messages according to the Upper Confidence bound applied to Trees rule:

$$\begin{aligned} m &= \arg\max_{m} Q(s,m) + c\sqrt{\frac{\ln \sum_{b} N(s,b)}{N(s,m)}} \\ Q(s,m) &= \frac{\sum_{i} G_{i}(s,m)}{N(s,m)} \end{aligned} \tag{6.7}$$

where $N(s,m)$ is the number of times message m was selected from the node, $G_i(s,m)$ are the returns obtained from the node when selecting m and c is a constant set to $\sqrt{2}$. When the architect must choose a message from the environment state s, its policy $\pi_{A}(m|s)$ runs the above procedure from a root node labeled with the current environment state s. After expanding a budget b of nodes the architect picks the best message to send according to Eq. (6.7) applied to the root node. It is then possible to reuse the tree for the next action selection or to discard it, if a tree is reused its maximal depth should be constrained. For more details on the MCTS procedure, the reader is referred to the Background, specifically Subsection 3.4.1.

Hyperparameters. All models are parametrized by two-hidden layer 126-units feedforward ReLU networks. Only the learning rates on BuildWorld were searched over with grid searches.

For BuildWorld with 3 blocks, the searched range is $[5 \times 10^{-4}, 1 \times 10^{-4}, 1 \times 10^{-5}]$ for both architect and builder (vocabulary size was fixed at 6). For 'grasp' with 6 blocks the searched range is $[1 \times 10^{-3}, 5 \times 10^{-4}, 1 \times 10^{-4}]$ for the architect and $[5 \times 10^{-4}, 1 \times 10^{-4}, 5 \times 10^{-5}]$ for the builder (vocabulary size was fixed at 72). The other hyperparameters do not seem to have a major impact on the performance provided that:

- the MCTS hyperparameters enable an agent that has access to the reward to solve the task.
- there is enough BC epochs to approach convergence.

Regarding the vocabulary size, the larger the better (see experiments in Figure 6.9). Sparse reward means that the architect receives 1 if the goal is achieved and 0 otherwise. Episodes per iteration are equally divided into the modeling and guiding frames. Hyperparameters are given in the tables below.

Compute. A complete ABIG training can take up to 48 hours on a single modern CPU (`Intel E5-2683 v4 Broadwell @ 2.1GHz`). The presented results require approximately 700 CPU hours. For each training, the main computation cost comes from the MCTS planning during the guiding frames. The self-imitation and behavior modeling steps only account for a small fraction of the computation.

Reproducibility and code. We ensure the reproducibility of the experiments presented in this work by providing our code (see `https://github.com/flowersteam/architect-builder-abig.git`). We also guarantee the statistical significance of our experimental results by using 10 random seeds, reporting the standard error of the mean, and using Welch's t-test.

budget	reuse tree	max tree depth
100	true	500

Table 6.1: MCTS parameters

episode len	grid size	reward	message
40	5×6 / (6 × 6)	sparse	one-hot

discount factor	episodes per iteration	vocab size	evaluation episode len
0.95	600	18 / (72)	40 / (60)

Table 6.2: BuildWorld parameters for 3 blocks / (for 6 blocks if different)

Algorithm 4 Architect-Builder Iterated Guiding (ABIG)

Require: randomly initialized builder policy π_{B}, reward function r, transition function P_{E}, BC algorithm, MCTS algorithm

for i in range($N_{iterations}$) **do**

▷ Modelling frame

for e in range($N_{collect}/2$) **do**

Architect populates $\mathcal{D}_{A}$ using $m \sim$ Uniform() and observing $a \sim \pi_{B}(\cdot|s, m)$

Architect learns $\tilde{\pi}_{B}(a|s, m)$ on $\mathcal{D}_{A}$ with BC

Architect sets $\pi_{A}(m|s) \triangleq \text{MCTS}(r, \tilde{\pi}_{B}, P_{E})$

Architect flushes $\mathcal{D}_{A}$

▷ Guiding frame

for e in range($N_{collect}/2$) **do**

Builder populates $\mathcal{D}_{B}$ using π_{B} while guided by Architect, i.e. $m \sim \pi_{A}(\cdot|s)$

Builder learns $\pi_{B}(a|s, m)$ on $\mathcal{D}_{B}$ with BC

Builder flushes $\mathcal{D}_{B}$

Architect runs one last Modelling Frame

return π_{A}, π_{B}

learning rate	number of epochs	batch-size	wait for
5×10^{-4}	1000	256	300

Table 6.3: Architect's BC parameters on BuildWorld for 3 blocks / (for 6 blocks if different)

learning rate	number of epochs	batch-size	wait for
1×10^{-4}	1000	256	300

Table 6.4: Builder's BC parameters on BuildWorld for 3 blocks / (for 6 blocks if different)

6.3 Understanding the learning mechanisms

Architect-Builder Iterated Guiding relies on two steps. First, the architect selects *favorable* messages, i.e., messages that maximize the likelihood of the builder picking optimal actions with respect to the architect's reward. Then, the builder does self-imitation and reinforces the guided behavior by maximizing the likelihood of the corresponding messages-actions sequence under its policy. The message-to-action associations (or preferences) are encoded in the builder's policy $\pi_{_B}(a|s,m)$. Maximum likelihood assumes that actions are initially equiprobable for a given message. Therefore, actions under a message that is not present in the dataset ($\mathcal{D}_{_B}$) remain so. In other words, if the builder never observes a message, it assumes that this message is equally associated with all the possible actions. This enables the builder to *forget* past message-to-action associations that are not used by the architect. In practice, initial uniform likelihood is ensured by resetting the builder's policy network before each self-imitation. The architect can leverage the forgetting mechanism to erase unfavorable associations until a favorable one emerges. Such favorable associations can then be reinforced by the architect-builder pair until it is made deterministic. The *reinforcement* process of favorable associations is also enabled by the self-imitation phase. Indeed, for a given message m, the self-imitation objective for π on a data-set $\mathcal{D}$ collected using π is:

$$J(m,\pi) = -\sum_{a\sim\mathcal{D}} \log \pi(a|m) \approx \mathbb{E}_{a\sim\pi(\cdot|m)}[-\log \pi(a|m)] \approx H[\pi(\cdot|m)] \tag{6.8}$$

where H stands for the entropy of a distribution. Therefore, maximizing the likelihood, in this case, results in minimizing the entropy of $\pi(\cdot|m)$ and thus reinforces the associations between messages and actions. Using these mechanisms, the architect can adjust the policy of the builder until it becomes *controllable*, i.e., deterministic (strong preferences over actions for a given message) and flexible (varied preferences across messages). Conversely, in the case of ABIG-no-intent, the architect does not guide the builder and simply sends messages at random. Favorable and unfavorable messages are thus sampled alike which prevents the forgetting mechanism to undo unfavorable message-to-action associations. Consequently, in that case, self-imitation tends to simply reinforce the initial builder's preferences over actions making the controllability of the builder policy depend heavily on the initial preferences.

We illustrate the above learning mechanisms in the following paragraph by applying ABIG to a simple instantiation of the ABP. Figures 6.4 and 6.6 confirm that ABIG uses the forgetting and reinforcement mechanisms to circumvent the unfavorable initial conditions while ABIG-no-intent simply reinforces them. Eventually, Figure 6.6 reports that ABIG always reaches 100% success rate regardless of the initial conditions while ABIG-no-intent success rate depends on the initial preferences (only 3% of success rate when they are unfavorable).

Interestingly, the emergent learning mechanisms discussed here are reminiscent of the amplification and self-enforcement of random fluctuations in naming games (Steels, 1995).

In naming games, however, the self-organization of vocabularies is driven by each agent maximizing its communicative success whereas in our case the builder has no external learning signal and simply self-imitates.

6.3.1 A detailed example

To illustrate the learning mechanisms of ABIG we propose to look at the simplest instantiation of the Architect-Builder Problem: there is one state (thus it can be ignored), two messages m_1 and m_2 and two possible actions a_1 and a_2. If the builder chooses a_1 it is a loss ($r(a_1) = -1$) but choosing a_2 results in a win ($r(a_2) = 1$). Figure 6.4 displays several iterations of ABIG on this problem when the initial builder's policy is unfavorable (a_1 is more likely than a_2 for all the messages). During each iteration, the architect selects messages in order to maximize the likelihood of the builder picking action a_2 and then the builder does self-Imitation Learning by maximizing the likelihood of the corresponding messages-actions sequence under its policy. Figure 6.4 shows that this process leads to forgetting unfavorable associations until a favorable association emerges and can be reinforced. On the other hand, for ABIG-no-intent in Figure 6.5, favorable and unfavorable messages are sampled alike which prevents the forget mechanism to undo unfavorable message-to-action associations. Consequently, initial preferences are reinforced.

To further assess how the architect's message choices impact the performance of a self-imitating builder, we compare the distribution of the builder's preferred actions obtained after using ABIG and ABIG-no-intent. We consider three different initial conditions (favorable, unfavorable, intermediate) that are each ran to convergence (meaning that the policy does not change anymore across iterations) for 100 different seeds. Figure 6.6 displays the resulting distributions of preferred – i.e., most likely – action for each message. When applying ABIG on the toy problem, the pair always reaches a success rate of 100/100 no matter the initial condition. We also observe that – at convergence – the builder never prefers action a_1, yet when an action is preferred for a given message, the other message yields no preference over action ($p(a_1|m) = p(a_2|m)$). This is due to the forgetting mechanism discussed previously. The results when applying ABIG-no-intent on the toy problem are much more dependent on the initial condition. In the unfavorable scenario, ABIG-no-intent fails heavily with only three seeds succeeding over the 100 experiments. This is due to the fact that, in absence of message guidance from the architect, the builder has a high chance to continually reinforce the association between the two messages and a_1, therefore losing. However, in rare cases, the builder can inverse the initial message-conditioned probabilities by 'luckily' sampling more often a_2 when receiving m_1 and win. This only happened three times over the 100 seeds. Finally, when initial conditions are more favorable, the self-imitation steps reinforce the association between the messages and a_2 which makes the builder prefer a_2 for at least one message and enables high success rates (100/100 for favorable and 98/100 for intermediate).

sampling temperature	samples per iteration	learning rate	number of epochs	batch size
0.5	100	0.1	1000	50

Table 6.5: Toy experiment hyperparameters

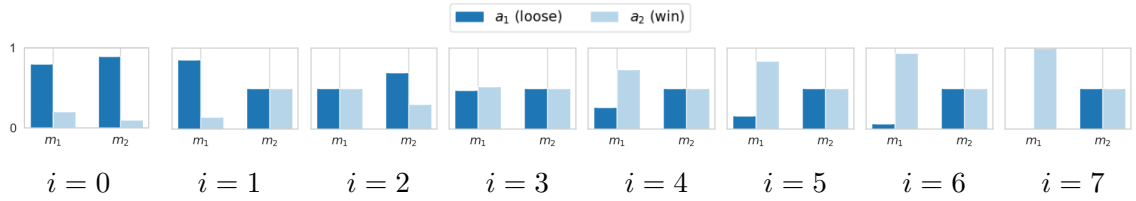


Figure 6.4: ABIG-driven evolution of message-conditioned action probabilities (builder's policy) for a simple problem where the builder must learn to produce action a_2. Even under unfavorable initial condition the architect-builder pair eventually manages to associate a message (here m_1) with the winning action (a_2). Initial conditions are unfavorable since a_1 is more likely than a_2 for both messages. ($i = 0$) Given the initial conditions, the architect only sends message m_1 since it is the most likely to result in action a_2. ($i = 1$) the builder guiding data only consisted of m_1 messages therefore it cannot learn a preference over actions for m_2 and both actions are equally likely under m_2. The architect now only sends message m_2 since it is more likely than m_1 at triggering a_2. ($i = 2$) Unfortunately, the sampling of m_1 resulted in the builder doing more a_1 than a_2 during the guiding frame and the builder thus associates m_2 with a_1. The architect tries its luck again but now with m_1. ($i = 3$) Eventually, the sampling results in more a_2 actions being sampled in the guiding data and the builder now associates m_1 to a_2. ($i = 4$) and ($i = 5$) The architect can now keep on sending m_1 messages to reinforce this association.

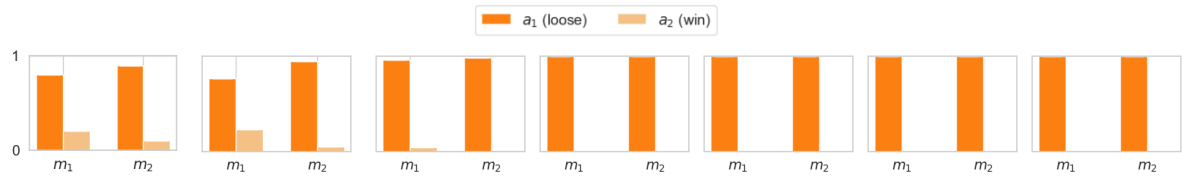
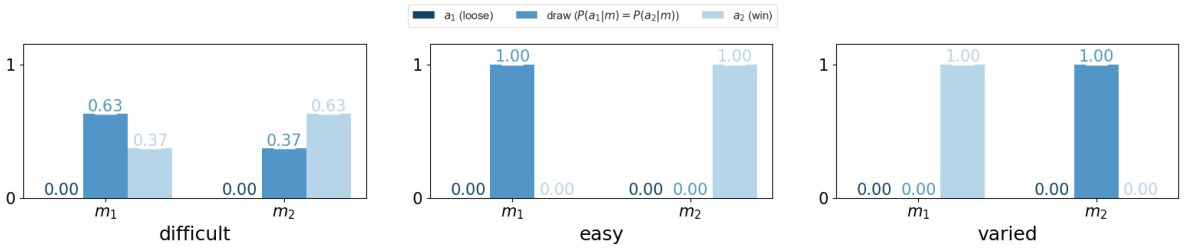


Figure 6.5: ABIG-no-intent driven evolution of message-conditioned action probabilities for a simple problem where the builder must learn to produce action a_2. Initial conditions are unfavorable since a_1 is more likely than a_2 for both messages. Without an architect's guiding messages during training, a self-imitating builder reinforces the action preferences of the initial conditions and fails (even when evaluated alongside a knowledgeable architect as both messages can only yield a_1).

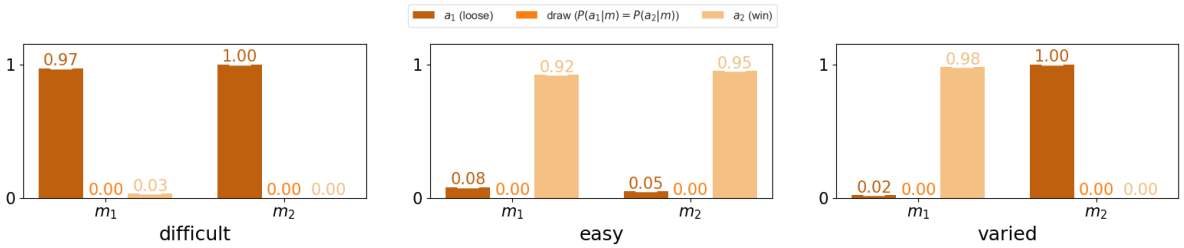
Unfavorable	Favorable	Intermediate
$P(a_1\|m_1) = 0.8$, $P(a_2\|m_1) = 0.2$ $P(a_1\|m_2) = 0.9$, $P(a_2\|m_2) = 0.1$	$P(a_1\|m_1) = 0.2$, $P(a_2\|m_1) = 0.8$ $P(a_1\|m_2) = 0.1$, $P(a_2\|m_2) = 0.9$	$P(a_1\|m_1) = 0.9$, $P(a_2\|m_1) = 0.1$ $P(a_1\|m_2) = 0.1$, $P(a_2\|m_2) = 0.9$

(a) **Initial probabilities**



Success Rate = 100/100 | Success Rate = 100/100 | Success Rate = 100/100

(b) **ABIG:** Distributions of final preferred action for each message



Success Rate = 3/100 | Success Rate = 100/100 | Success Rate = 98/100

(c) **ABIG-no-intent:** Distributions of final preferred action for each message

Figure 6.6: Toy experiment analysis (a) Initial conditions: initial probability for each action a given a message m; distributions of final builder's preferred actions for each message after applying (b) ABIG and (c) ABIG-no-intent on the toy problem; distributions are calculated over 100 seeds. For each method and each initial condition, we report the success rate obtained by a knowledgeable architect guiding the builder. At evaluation, the architect has access to the builder's model and does not ignore the goal. ABIG always succeeds while ABIG-no-intent's success depends on the initial conditions.

6.4 Results

In the following sections, success rates (sometimes referred to as scores) are averaged over 10 random seeds and error bars are ±2SEM. If not stated otherwise, the grid size is (5×6), contains three blocks ($N_b = 3$) and the vocabulary size is $|\mathcal{V}| = 18$.

6.4.1 ABIG's learning performances

We apply ABIG to the four learning tasks of BuildWorld and compare it with the two control settings: ABIG-no-intent (no guiding during training) and random (builder takes random actions). Figure 6.7 reports the mean success rate on the four tasks defined in Section 6.1. First, we observe that ABIG significantly outperforms the control conditions on all tasks. Second, we notice that on the simpler 'grasp' task ABIG-no-intent achieves a satisfactory mean score of 0.77±0.03. This is consistent with the learning mechanisms analysis provided in Subsection 6.3.1 that shows that, in favorable settings, a self-imitating builder can develop a reasonably controllable policy even if it learns on non-guiding trajectories. Nevertheless, when the tasks get more complicated and involve placing objects or drawing lines, the performances of ABIG-no-intent drop significantly whereas ABIG continues to achieve high success rates (> 0.8). This demonstrates that ABIG enables a builder-architect pair to successfully agree on a communication protocol that makes the builder's policy controllable and enables the architect to efficiently guide it.

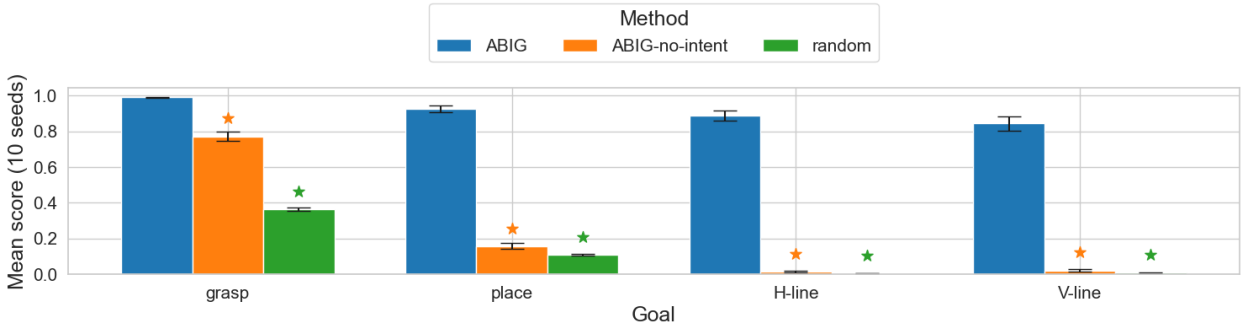


Figure 6.7: Methods performances. Stars indicate significance with respect to ABIG model according to Welch's t-test with null hypothesis $\mu_1 = \mu_2$, at level $\alpha = 0.05$. ABIG outperforms control baselines on all goals.

To highlight the need for a builder's policy that is controllable (i.e., both deterministic and flexible), we define two extra baselines:

- stochastic: where the builder policy is a fixed softmax policy parameterized by a randomly initialized network, and

- deterministic: where the builder policy is a fixed argmax policy parameterized by a randomly initialized network.

To even further simplify the architect's task, we give it direct access to the exact policy of the builder ($\tilde{\pi}_B = \pi_B$). It can thus use it without approximation to plan and guide the builder during evaluation. In the performances reported in Figure 6.8, the stochastic condition exhibits similar performances as the random builder. This indicates that, even if the architect tries to guide the builder, the stochastic policy is not controllable and performances are not improved. Finally, we would expect a deterministic policy to be more easily controllable by the architect. Yet, as pointed out in Figure 6.8, the initial deterministic policies lack flexibility and fail. This shows that the builder must iteratively co-evolve its policy in order to make it controllable by the architect.

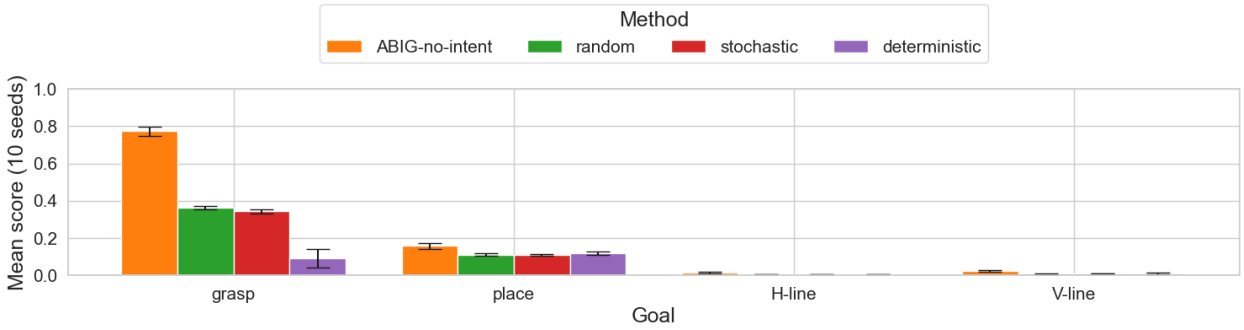


Figure 6.8: Baseline performance depending on the goal. Stochastic policy behaves on par with the random builder. Self-imitation with ABIG-no-intent remains the most controllable baseline.

Figure 6.9 shows ABIG's performance increasing with the vocabulary size, suggesting that with more messages available, the architect can more efficiently refer to the desired action. This type of relationship between vocabulary size, capacity, and performance echoes works that investigate compositionality and generalization in emergent languages (Chaabouni et al., 2020).

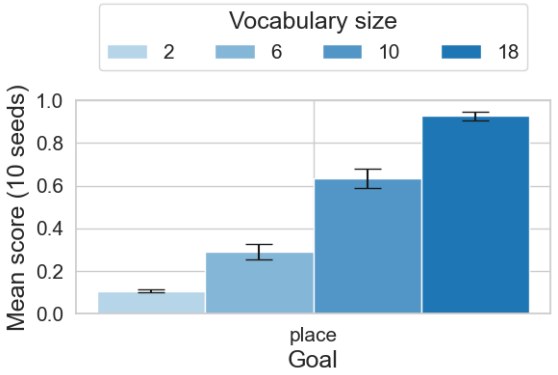


Figure 6.9: Influence of the vocabulary size. Experiments for ABIG on the 'place' task. Performance increases with the vocabulary size.

6.4.2 ABIG's transfer performances

Building upon previous results, we propose to study whether a learned communication protocol can transfer to new tasks. The architect-builder pairs are trained on a single task and then evaluated without retraining on the four tasks. In addition, we include 'all-goals': a control setting in which the builder learns a single policy by being guided on all four goals during training. Figure 6.9 shows that, on all training tasks except 'grasp', ABIG enables a transfer performance above 0.65 on all testing tasks. Notably, training on 'place' results in a robust communication protocol that can be used to solve the other tasks with a success rate above 0.85, being effectively equivalent to training on 'all-goals' directly. This might be explained by the fact that placing blocks at specified locations is an atomic operation required to build lines.

Figure 6.9: ABIG transfer performances. For different training goals and without retraining. ABIG agents learn a communication protocol that transfers to new tasks. The highest performances are reached when training on 'place'.

Challenging ABIG's transfer abilities. Motivated by ABIG's transfer performances, we propose to train it on the 'place' task in a bigger grid (6×6) with $N_b = 6$ and $|\mathcal{V}| = 72$. Then, without retraining, we evaluate it on the '6-block-shapes' task[2] that consists of constructing the shapes given in Figure 6.10. The training performance on 'place' is 0.96 ± 0.02 and the transfer performance on the '6-block-shapes' is 0.85 ± 0.03 (see Table 6.6). This further demonstrates ABIG's ability to derive robust communication protocols that can solve more challenging unseen tasks.

	Training ('place')	Transfer ('6-block-shapes')
Mean Score	0.96 ± 0.02	0.85 ± 0.03

Table 6.6: Transfer to complex tasks. The communication protocol learned on the simple 'place' task transfers well to more complex structure constructions.

[2]For rollouts see `https://sites.google.com/view/architect-builder-problem/`

Figure 6.10: 6-block-shapes that ABIG can construct in transfer mode when trained on the 'place' task.

6.4.3 ABIG's learning dynamics

We have shown performances (success rates) of builder-architect pairs at convergence. In this subsection, we propose to thoroughly study the evolution of the builder's policy in order to provide a deeper analysis of ABIG's learning dynamics.

Metric definition. We define three metrics that characterize the builder's behavior. We compute these metrics on a constant measurement set $\mathcal{M}$ made of 6000 randomly sampled states, for each of these states we sample all the possible messages $m \sim \text{Uniform}(\mathcal{V})$ where $\mathcal{V}$ is the set of possible messages. Therefore, $|\mathcal{M}| = 6000 \times |\mathcal{V}|$. The set of possible actions is $\mathcal{A}$ and we denote by δ the indicator function. We also define the following distributions:

$$p_S(s) \triangleq \frac{1}{|\mathcal{M}|} \sum_{\mathcal{M}} \delta(s' = s)$$

$$p_M(m) \triangleq P(m|s) = \frac{1}{|\mathcal{V}|}, \quad p_{SM}(s,m) \triangleq p_S(s)P(m|s) = p_S(s)p_M(m)$$

$$p_{SMA}(s,m,a) \triangleq p_{SM}(s,m)P(a|s,m) = p_{SM}(s,m)\pi_B(a|s,m)$$

$$p_A(a) \triangleq \sum_{\mathcal{M}} p_{SMA}(s,m,a), \quad p_{MA}(m,a) \triangleq \sum_{\mathcal{M}} p_{SMA}(s,m,a), \quad p_{SA}(s,a) \triangleq \sum_{\mathcal{M}} p_{SMA}(s,m,a)$$

From this we can define the following monitoring metrics:

- *Mean Entropy*

$$\bar{H}(\pi) = \frac{1}{|\mathcal{M}|} \sum_{\mathcal{M}} \left[-\sum_{\mathcal{A}} \pi(a|s,m) \log \pi(a|s,m) \right]$$

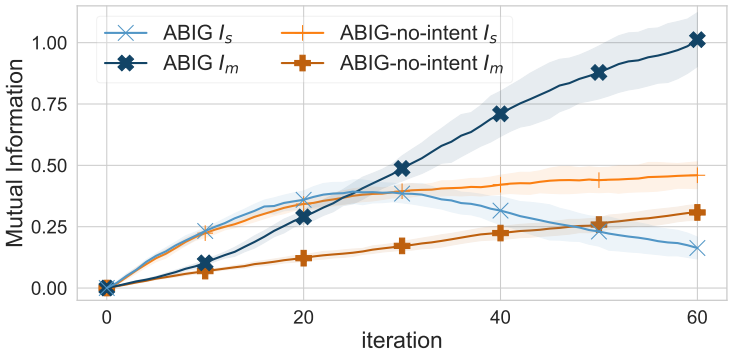
- *Mutual Information between messages and actions*

$$I_m = \sum_{\mathcal{V}} \sum_{\mathcal{A}} p_{MA}(m,a) \log \frac{p_{MA}(m,a)}{p_A(a)p_M(m)}$$

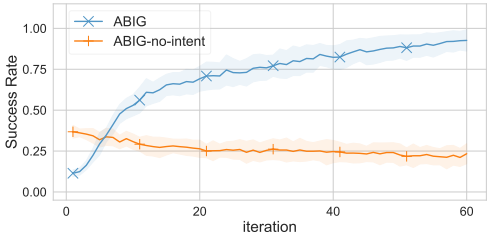
- *Mutual Information between states and actions*

$$I_s = \sum_{\mathcal{M}} \sum_{\mathcal{A}} p_{SA}(s,a) \log \frac{p_{SA}(s,a)}{p_A(a)p_S(s)}$$

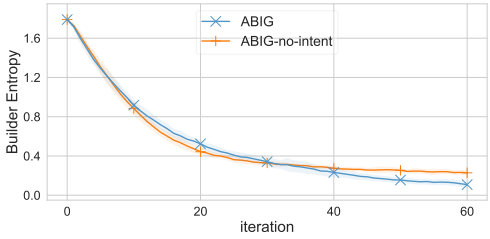
Analysis. Figure 6.11 displays the evolution of these metrics after each iteration as well as the evolution of the success rate (b). As indicated by Eq. (6.8), doing self-imitation learning results in a decay of the mean entropy (c). This decay is similar for ABIG and ABIG-no-intent. The most interesting result is provided by the evolution of the mutual information (a). For ABIG-no-intent, we see that I_s and I_m slowly increase with $I_s > I_m$ over all iterations. This indicates that the builder policy $\pi_B(a|s, m)$ relies more on states than on messages to compute the actions. In this scenario, the builder, therefore, tends to ignore messages. On the other hand, I_s and I_m evolve differently for ABIG. Both metrics first increase with $I_s > I_m$ until they cross around iteration 25. Then I_s starts decreasing and I_m grows. This shows that, as the communication protocol settles, ABIG increases the mutual information of message-action over the state-action one and results in a builder that selects actions based on the messages it receives. This is a very desirable feature for the emergence of communication.



(a) Evolution of the mutual information I_s and I_m



(b) Evolution of the success rate



(c) Evolution of the builder policy mean entropy $\bar{H}_{\pi_B}$

Figure 6.11: Comparison of the evolution of builder policy properties when applying ABIG and ABIG-no-intent on the 'place' task in BuildWorld. (a) ABIG promotes the mutual information between messages and actions which indicates successful communication protocols. (b) ABIG enables much higher performance that ABIG-no-intent. (c) Both methods use self-imitation and thus reduce the entropy of the policy.

6.5 Discussion

This work is inspired by experimental semiotics (Galantucci and Garrod, 2011) and in particular Vollmer et al. (2014) that studied the CoCo game with human participants as a key step towards understanding the underlying mechanisms of coordination for the emergence of communication. Here we take a complementary approach by defining and investigating solutions to the Architect-Builder Problem, a general formulation of the CoCo game where both agents are AIs.

Recent MARL works investigate how RL agents trained in the presence of other agents leverage the behaviors they observe to improve learning (Lowe et al., 2017; Woodward et al., 2020; Roy et al., 2020; Ndousse et al., 2021). In these settings, the other agents are used to build useful representations or gain task-related information, yet, the main learning signal of every agent remains the ground truth reward.

Feudal Learning investigates a setting where a manager sets the rewards of workers to maximize its own return (Dayan and Hinton, 1993; Kulkarni et al., 2016; Vezhnevets et al., 2017; Nachum et al., 2018a; Ahilan and Dayan, 2019). In these Hierarchical frameworks, the manager interacts by directly tweaking the workers' learning signal. For instance, Kulkarni et al. (2016) propose to decompose an RL agent into a two-stage hierarchy with a meta-controller (the manager) setting the goals of a controller (the worker). The meta-controller is trained to select sequences of goals that maximize the environment reward while the controller is trained to maximize goal-conditioned intrinsic rewards. The definition of the goal space as well as the corresponding hard-coded goal-conditioned reward functions are task-related design choices. Vezhnevets et al. (2017) propose a more general approach by defining goals as embeddings that directly modulate the worker's policy. Additionally, the authors define intrinsic rewards as the cosine distance between goals and embedded state deltas (the difference between the embedded state at the moment the goal was given and the current embedded state). Thus, goals can be interpreted as directions in embedding space. Nachum et al. (2018a) simplify this idea by letting go of the embedding transformation and considering goals as directions to reach, and rewards as distances between state deltas and goals. These works tackle the single-agent learning problem and therefore allow the manager to directly influence the learning signal of the workers. This would be unfeasible in the multi-agent setting where agents are physically distinct and it is not possible for an agent to explicitly tweak another agent's learning algorithm. Instead, agents must communicate by influencing each other's *observations* instead of intrinsic *rewards*. Designed to investigate the emergence of communication, ABP lies in this latter multi-agent setting where separate entities can only interact with one another through communications and influence each other's observations instead of reward signals. Additionally, Feudal or Hierarchical methods require worker agents that directly receive and maximize rewards. In contrast, in ABP, the builder is reward-less and observes communication messages (which, initially, have arbitrary meanings) to figure out what it should do.

Inverse Reinforcement Learning and Imitation Learning have been investigated for Human-Robot Interaction when it is challenging to specify a reward function (Pomerleau, 1991; Ng et al., 2000). Instead of defining rewards, IRL and IL rely on expert demonstrations. Hadfield-Menell et al. (2016) argue that learning from expert demonstrations is not always optimal and investigate how to produce instructive demonstrations to best teach an apprentice. Crucially, the expert is aware of the mechanisms by which the apprentice learns, namely RL on top of IRL. This allows the expert to assess how its demonstrations influence the apprentice policy, effectively reducing the problem to a single agent POMDP. In our case, however, the architect and the builder do not share the same action space which prevents the architect from producing demonstrations. In addition, the architect has no knowledge of the builder's learning process which makes the simplification to a single-agent teacher problem impossible.

In essence, the ABP is closest to works tackling the calibration-free Brain-Computer Interface control problem (Grizou et al., 2014a; Xie et al., 2021). Yet, these works both consider that the architect sends messages after the builder's actions and thus enforce that the feedback conveys a reward. Crucially, the architect does not learn, rather, it communicates with a fixed mapping between feedback and pre-defined meanings ("correct" vs. "wrong"). These meanings are known to the builder and this latter simply has to learn the mapping between feedback and meaning. In our case, however, the architect communicates before the builder's action and thus gives instructions rather than feedback. Additionally, the builder has no a priori knowledge of the set of possible meanings and the architect adapts these latter to the builder's reaction. Finally, Grizou et al. (2013) handles both feedback and instruction communications but relies on known task distribution and a set of possible meanings. In terms of motivations, previous works are interested in one robot figuring out a fixed communication protocol while we train two agents to collectively emerge one.

Our BuildWorld resembles GridLU proposed by Bahdanau et al. (2019) to analyze reward modeling in language-conditioned learning. However, their setting is fundamentally different from ours as it investigates single-agent goal-conditioned IL where goals are predefined episodic linguistic instructions labeling expert demonstrations. Nguyen et al. (2021) alleviate the need for expert demonstrations by introducing an interactive teacher that provides descriptions of the learning agent's trajectories. In this HRI setting, the teacher still follows a fixed pre-defined communication protocol known by the learner: messages are activity descriptions. Our ABP formulation relates to the Minecraft Collaborative Building Task and the IGLU competition (Narayan-Chen et al., 2019; Kiseleva et al., 2021); however, they do not consider emergent communication. Rather, they focus on generating architect utterances by leveraging a human-human dialogues corpus to learn pre-established meanings expressed in natural language. Conversely, in ABP both agents learn and must evolve the meanings of messages while solving the task without relying on any form of demonstration.

This work formalizes the ABP as an interactive setting where learning and coordination must occur without explicit reinforcement, demonstrations, or a shared language. This setting is best suited to tackle the calibration-free BCI problem in which a user (the architect) tries to control a prosthetic arm (the builder) from brain signals (the messages) alone. Since there are no general and robust *a-priori* mappings between a user's brain signals and what the arm should do, it might be best for the user and the prothese to coordinate by interactively defining this communication protocol together.

To address the ABP, we propose ABIG, an algorithm that allows us to learn how to guide and to be guided. ABIG is only based on two high-level priors to the emergence of communication: shared intent and interaction frames. ABP's general formulation enables us to formally enforce those priors during learning. We study their influence through ablation studies, highlighting the importance of shared intent achieved by doing self-imitation on guiding trajectories. When performed across interaction frames, this mechanism enables agents to efficiently evolve a communication protocol that allows them to solve all the tasks defined in BuildWorld. More impressively, we find that communication protocols derived from a simple task can be used to solve harder, never-seen goals.

Chapter 7

Learning better internal models of others

We have seen throughout the literature review and the works presented in this thesis that coordination in multi-agent learning heavily relies on being able to build accurate models of other agents. Currently, most works rely on the basic Behavioral Cloning approach to learn these approximate models since more sophisticated methods, such as the ones leveraging adversarial learning, are complex, computationally intensive, brittle, and require extensive hyperparameter tuning. This prevents their use to model teammates in multi-agent learning since, in addition to optimizing its own policy, each agent would have to learn a complex internal model for every other agent. Building from this observation, this chapter features the work that we published in Barde et al. (2020) and proposes a simple, practical, and robust Imitation Learning method that is competitive to the much more complex, yet prevalent, Adversarial Imitation Learning (AIL) approaches.

Specifically, AIL alternates between learning a discriminator – which tells apart expert's demonstrations from generated ones – and a generator's policy to produce trajectories that can fool this discriminator. This alternated optimization is known to be delicate in practice since it compounds unstable adversarial training with brittle and sample-inefficient reinforcement learning. We propose to remove the burden of the policy optimization steps by leveraging a novel discriminator formulation. Specifically, our discriminator is explicitly conditioned on two policies: the one from the previous generator's iteration and a learnable policy. When optimized, this discriminator directly learns the optimal generator's policy. Consequently, our discriminator's update solves the generator's optimization problem for free: learning a policy that imitates the expert does not require an additional optimization loop. This formulation effectively cuts by half the implementation and computational burden of Adversarial Imitation Learning algorithms by removing the Reinforcement Learning phase altogether. We show on a variety of tasks that our simpler approach is competitive to

the popular IL methods.

Imitation Learning treats the task of learning a policy from a set of expert demonstrations. IL is effective on control problems that are challenging for traditional RL methods, either due to reward function design challenges or the inherent difficulty of the task itself (Abbeel and Ng, 2004; Ross et al., 2011). Since it enables learning policies from state-action trajectories, it also provides a powerful tool to model other agents from the behaviors they demonstrate.

Most IL work can be divided into two branches: Behavioral Cloning and Inverse Reinforcement Learning. BC casts IL as a supervised learning objective and seeks to imitate the expert's actions using the provided demonstrations as a fixed dataset (Pomerleau, 1991). Thus, BC usually requires a lot of expert data and results in agents that struggle to generalize. As an agent deviates from the demonstrated behaviors – straying outside the state distribution on which it was trained – the risks of making additional errors increase, a problem known as compounding error (Ross et al., 2011).

Inverse Reinforcement Learning aims to reduce compounding error by learning a reward function under which the expert policy is optimal (Abbeel and Ng, 2004). Once learned, an agent can be trained (with any RL algorithm) to learn how to act at any given state of the environment. Early methods were prohibitively expensive in large environments because they required training the RL agent to convergence at each learning step of the reward function (Ziebart et al., 2008; Abbeel and Ng, 2004). Recent Adversarial Imitation Learning approaches instead apply an adversarial formulation in which a discriminator learns to distinguish between expert and agent behaviors to learn the reward optimized by the expert. AIL methods allow for the use of function approximators and can in practice be used with only a few policy improvement steps for each discriminator update (Ho and Ermon, 2016; Fu et al., 2017; Finn et al., 2016a).

While these advances have allowed IL to tackle bigger and more complex environments (Kuefler et al., 2017; Ding et al., 2019), they have also significantly complexified the implementation and learning dynamics of the algorithms. It is worth asking how much of this complexity is actually mandated. For example, in recent work, Reddy et al. (2019) have shown that competitive performance can be obtained by hard-coding a very simple reward function to incentivize expert-like behaviors and manage to imitate it through off-policy direct RL. Reddy et al. (2019) therefore remove the reward learning component of AIL and focus on the RL loop, yielding a regularized version of BC. Motivated by these results, we also seek to simplify the AIL framework but follow the opposite direction: keeping the reward learning module and removing the policy improvement loop.

We propose a simpler yet competitive AIL framework. Motivated by Finn et al. (2016a) who use the optimal discriminator form, we propose a structured discriminator that estimates the probability of demonstrated and generated behavior using a single parameterized maximum entropy policy. Discriminator learning and policy learning, therefore, occur simultaneously, rendering seamless generator updates: once the discriminator has been trained for

a few epochs, we simply use its policy model to generate new rollouts. We call this approach Adversarial Soft Advantage Fitting (ASAF).

We make the following contributions:

- **Algorithmic**: we present a novel algorithm (ASAF) designed to imitate expert demonstrations without any Reinforcement Learning step.
- **Theoretical**: we show that our method retrieves the expert policy when trained to optimality.
- **Empirical**: we show that ASAF outperforms prevalent IL algorithms on a variety of discrete and continuous control tasks. We also show that, in practice, ASAF can be easily modified to account for different trajectory lengths (from full length to transition-wise).

7.1 Imitation Learning without Policy Optimization

In this section, we derive Adversarial Soft Advantage Fitting, our novel Adversarial Imitation Learning approach. Specifically, in Section 7.1.1, we present the theoretical foundations for ASAF to perform IL on full-length trajectories. Intuitively, our method is based on the use of such *structured discriminators* – that match the optimal discriminator form – to fit the trajectory distribution induced by the expert policy. This approach requires being able to evaluate and sample from the learned policy and allows us to learn that policy and train the discriminator simultaneously, thus drastically simplifying the training procedure. We present in Section 7.1.2 parametrization options that satisfy these requirements. Finally, in Section 7.1.3, we explain how to implement a practical algorithm that can be used for arbitrary trajectory lengths, including the transition-wise case.

7.1.1 Adversarial Soft Advantage Fitting – theoretical setting

Before introducing our method, we derive GAN training with a structured discriminator.

GAN with structured discriminator. Suppose that we have a generator distribution p_G and some arbitrary distribution $\tilde{p}$ and that both can be evaluated efficiently, e.g., categorical distribution or probability density with normalizing flows (Rezende and Mohamed, 2015). We call a *structured discriminator* a function $D_{\tilde{p},p_G} : \mathcal{X} \to [0,1]$ of the form $D_{\tilde{p},p_G}(x) = \tilde{p}(x) \big/ (\tilde{p}(x) + p_G(x))$ which matches the optimal discriminator form for Eq. (3.42). Considering our new GAN objective, we get:

$$\min_{p_G} \max_{\tilde{p}} L(\tilde{p}, p_G)\,, \quad L(\tilde{p}, p_G) \triangleq \mathbb{E}_{x\sim p_E}[\log D_{\tilde{p},p_G}(x)] + \mathbb{E}_{x\sim p_G}[\log(1 - D_{\tilde{p},p_G}(x))]. \tag{7.1}$$

While the unstructured discriminator D from Eq. (3.41) learns a mapping from x to a Bernoulli distribution, we now learn a mapping from x to an arbitrary distribution $\tilde{p}$

from which we can analytically compute $D_{\tilde{p},p_G}(x)$. One can therefore say that $D_{\tilde{p},p_G}$ is *parameterized* by $\tilde{p}$. For the optimization problem of Eq. (7.1), we have the following optima:

Lemma 1. *The optimal discriminator parameter for any generator p_G in Eq. (7.1) is equal to the expert's distribution, $\tilde{p}^* \triangleq \arg\max_{\tilde{p}} L(\tilde{p}, p_G) = p_E$, and the optimal discriminator parameter is also the optimal generator, i.e.,*

$$p_G^* \triangleq \arg\min_{p_G} \max_{\tilde{p}} L(\tilde{p}, p_G) = \arg\min_{p_G} L(p_E, p_G) = p_E = \tilde{p}^*.$$

Proof. Lemma 1 states that given $L(\tilde{p}, p_G)$ defined in Eq. (7.1):

(a) $\tilde{p}^* \triangleq \arg\max_{\tilde{p}} L(\tilde{p}, p_G) = p_E$

(b) $\arg\min_{p_G} L(p_E, p_G) = p_E$

Starting with (a), we have:

$$\begin{aligned}\arg\max_{\tilde{p}} L(\tilde{p}, p_G) &= \arg\max_{\tilde{p}} \sum_{x_i} p_E(x_i) \log D_{\tilde{p},p_G}(x_i) + p_G(x_i) \log(1 - D_{\tilde{p},p_G}(x_i)) \\ &\triangleq \arg\max_{\tilde{p}} \sum_{x_i} L_i\end{aligned}$$

Assuming infinite discriminator's capacity, L_i can be made independent for all $x_i \in \mathcal{X}$ and we can construct our optimal discriminator $D^*_{\tilde{p},p_G}$ as a look-up table $D^*_{\tilde{p},p_G} : \mathcal{X} \to]0,1[\,;\, x_i \mapsto D_i^*$ with D_i^* the optimal discriminator for each x_i defined as:

$$D_i^* = \arg\max_{D_i} L_i = \arg\max_{D_i} p_{E,i} \log D_i + p_{G,i} \log(1 - D_i), \tag{7.2}$$

with $p_{G,i} \triangleq p_G(x_i)$, $p_{E,i} \triangleq p_E(x_i)$ and $D_i \triangleq D(x_i)$.

Recall that $D_i \in]0,1[$ and that $p_{G,i} \in]0,1[$. Therefore the function $\tilde{p}_i \mapsto D_i = \dfrac{\tilde{p}_i}{\tilde{p}_i + p_{G,i}}$ is defined for $\tilde{p}_i \in]0, +\infty[$. Since it is strictly monotonic over that domain we have that:

$$D_i^* = \arg\max_{D_i} L_i \;\Leftrightarrow\; \tilde{p}_i^* = \arg\max_{\tilde{p}_i} L_i \tag{7.3}$$

Taking the derivative and setting to zero, we get:

$$\left.\frac{dL_i}{d\tilde{p}_i}\right|_{\tilde{p}_i} = 0 \;\Leftrightarrow \tilde{p}_i = p_{E,i} \tag{7.4}$$

The second derivative test confirms that we have a maximum, i.e. $\left.\frac{d^2 L_i}{d\tilde{p}_i^2}\right|_{\tilde{p}_i^*} < 0$. The values of L_i at the boundaries of the domain of definition of $\tilde{p}_i$ tend to $-\infty$, therefore $L_i(\tilde{p}_i^* = p_{E,i})$ is the global maximum of L_i w.r.t. $\tilde{p}_i$. Finally, the optimal global discriminator is given by:

$$D^*_{\tilde{p},p_G}(x) = \frac{p_E(x)}{p_E(x) + p_G(x)} \quad \forall x \in \mathcal{X} \tag{7.5}$$

This concludes the proof for (a).

The proof for (b) can be found in the work of Goodfellow et al. (2014). We reproduce it here for completion. Since from (a) we know that $\tilde{p}^*(x) = p_E(x) \, \forall x \in \mathcal{X}$, we can write the GAN objective for the optimal discriminator as:

$$\begin{aligned} \arg\min_{p_G} L(\tilde{p}^*, p_G) &= \arg\min_{p_G} L(p_E, p_G) \\ &= \arg\min_{p_G} \mathbb{E}_{x\sim p_E}\left[\log \frac{p_E(x)}{p_E(x) + p_G(x)}\right] + \mathbb{E}_{x\sim p_G}\left[\log \frac{p_G(x)}{p_E(x) + p_G(x)}\right] \end{aligned} \tag{7.6}$$

Note that:

$$\log 4 = \mathbb{E}_{x\sim p_E}\left[\log 2\right] + \mathbb{E}_{x\sim p_G}\left[\log 2\right] \tag{7.7}$$

Adding Eq. (7.7) to Eq. (7.6) and subtracting $\log 4$ on both sides:

$$\begin{aligned} \arg\min_{p_G} L(p_E, p_G) &= -\log 4 + \mathbb{E}_{x\sim p_E}\left[\log \frac{2p_E(x)}{p_E(x) + p_G(x)}\right] + \mathbb{E}_{x\sim p_G}\left[\log \frac{2p_G(x)}{p_E(x) + p_G(x)}\right] \\ &= -\log 4 + D_{\mathrm{KL}}\left(p_E \middle\| \frac{p_E + p_G}{2}\right) + D_{\mathrm{KL}}\left(p_E \middle\| \frac{p_E + p_G}{2}\right) \\ &= -\log 4 + 2D_{\mathrm{JS}}\left(p_E \| p_G\right) \end{aligned} \tag{7.8}$$

Where D_{KL} and D_{JS} are respectively the Kullback–Leibler and the Jensen-Shannon divergences. Since the Jensen-Shannon divergence between two distributions is always non-negative and zero if and only if the two distributions are equal, we have that $\arg\min_{p_G} L(p_E, p_G) = p_E$. This concludes the proof for (b).

□

Intuitively, Lemma 1 shows that the optimal discriminator parameter is also the target data distribution of our optimization problem (i.e., the optimal generator). In other words, solving the inner optimization yields the solution of the outer optimization. In practice, we update $\tilde{p}$ to minimize the discriminator objective and use it directly as p_G to sample new data.

Matching trajectory distributions with a structured discriminator. Motivated by the GAN with a structured discriminator, we consider the trajectory distribution matching problem in IL. Here, we optimise Eq. (7.1) with $x = \tau, \mathcal{X} = \mathcal{T}, p_{_E} = P_{\pi_{_E}}, p_{_G} = P_{\pi_{_G}}$, which yields the following objective:

$$\min_{\pi_{_G}} \max_{\tilde{\pi}} L(\tilde{\pi}, \pi_{_G}), \quad L(\tilde{\pi}, \pi_{_G}) \triangleq \mathbb{E}_{\tau \sim P_{\pi_{_E}}}[\log D_{\tilde{\pi}, \pi_{_G}}(\tau)] + \mathbb{E}_{\tau \sim P_{\pi_{_G}}}[\log(1 - D_{\tilde{\pi}, \pi_{_G}}(\tau))], \tag{7.9}$$

with the structured discriminator:

$$D_{\tilde{\pi}, \pi_{_G}}(\tau) = \frac{P_{\tilde{\pi}}(\tau)}{P_{\tilde{\pi}}(\tau) + P_{\pi_{_G}}(\tau)} = \frac{q_{\tilde{\pi}}(\tau)}{q_{\tilde{\pi}}(\tau) + q_{\pi_{_G}}(\tau)}. \tag{7.10}$$

Here we used the fact that $P_\pi(\tau)$ decomposes into two distinct products:

$$q_\pi(\tau) \triangleq \prod_{t=0}^{T-1} \pi(a_t|s_t)$$

which depends on the stationary policy π, and

$$\xi(\tau) \triangleq \mathcal{P}_0(s_0) \prod_{t=0}^{T-1} \mathcal{P}(s_{t+1}|s_t, a_t)$$

which accounts for the environment dynamics. Crucially, $\xi(\tau)$ cancels out in the numerator and denominator leaving $\tilde{\pi}$ as the sole parameter of this structured discriminator. In this way, $D_{\tilde{\pi}, \pi_{_G}}(\tau)$ can evaluate the probability of a trajectory being generated by the expert policy simply by evaluating products of stationary policy distributions $\tilde{\pi}$ and $\pi_{_G}$. With this form, we can get the following result:

Theorem 1. *The optimal discriminator parameter for any generator policy $\pi_{_G}$ in Eq. (7.9) $\tilde{\pi}^* \triangleq \arg\max_{\tilde{\pi}} L(\tilde{\pi}, \pi_{_G})$ is such that $q_{\tilde{\pi}^*} = q_{\pi_{_E}}$, and using generator policy $\tilde{\pi}^*$ minimizes $L(\tilde{\pi}^*, \pi_{_G})$, i.e.,*

$$\tilde{\pi}^* \in \arg\min_{\pi_{_G}} \max_{\tilde{\pi}} L(\tilde{\pi}, \pi_{_G}) = \arg\min_{\pi_{_G}} L(\tilde{\pi}^*, \pi_{_G}).$$

Proof. Theorem 1 states that given $L(\tilde{\pi}, \pi_{_G})$ defined in Eq. (7.9):

(a) $\tilde{\pi}^* \triangleq \arg\max_{\tilde{\pi}} L(\tilde{\pi}, \pi_{_G})$ satisfies $q_{\tilde{\pi}^*} = q_{\pi_{_E}}$

(b) $\pi_{_G}^* = \tilde{\pi}^* \in \arg\min_{\pi_{_G}} L(\tilde{\pi}^*, \pi_{_G})$

The proof of (a) is very similar to the one from Lemma 1. Starting from Eq. (7.9) we

have:

$$\begin{aligned}\arg\max_{\tilde{\pi}} L(\tilde{\pi}, \pi_{_G}) &= \arg\max_{\tilde{\pi}} \sum_{\tau_i} P_{\pi_{_E}}(\tau_i) \log D_{\tilde{\pi},\pi_{_G}}(\tau_i) + P_{\pi_{_G}}(\tau_i) \log(1 - D_{\tilde{\pi},\pi_{_G}}(\tau_i)) \\ &= \arg\max_{\tilde{\pi}} \sum_{\tau_i} \xi(\tau_i) \left(q_{\pi_{_E}}(\tau_i) \log D_{\tilde{\pi},\pi_{_G}}(\tau_i) + q_{\pi_{_G}}(\tau_i) \log(1 - D_{\tilde{\pi},\pi_{_G}}(\tau_i)) \right) \\ &= \arg\max_{\tilde{\pi}} \sum_{\tau_i} L_i \end{aligned} \tag{7.11}$$

Like for Lemma 1, we can optimize for each L_i individually. When doing so, $\xi(\tau_i)$ can be omitted as it is constant w.r.t $\tilde{\pi}$. The rest of the proof is identical to the one of but Lemma 1 with $p_{_E} = q_{\pi_{_E}}$ and $p_{_G} = q_{\pi_{_G}}$. It follows that the max of $L(\tilde{\pi}, \pi_{_G})$ is reached for $q^*_{\tilde{\pi}} = q_{\pi_{_E}}$. From that we obtain that the policy $\tilde{\pi}^*$ that makes the discriminator $D_{\tilde{\pi}^*,\pi_{_G}}$ optimal w.r.t $L(\tilde{\pi}, \pi_{_G})$ is such that $q_{\tilde{\pi}^*} = q^*_{\tilde{\pi}} = q_{\pi_{_E}}$ i.e. $\prod_{t=0}^{T-1} \tilde{\pi}^*(a_t|s_t) = \prod_{t=0}^{T-1} \pi_{_E}(a_t|s_t) \, \forall \, \tau$.

The proof for (b) stems from the observation that choosing $\pi_{_G} = \tilde{\pi}^*$ (the policy recovered by the optimal discriminator $D_{\tilde{\pi}^*,\pi_{_G}}$) minimizes $L(\tilde{\pi}^*, \pi_{_G})$:

$$\begin{aligned} \pi_{_G}(a|s) = \tilde{\pi}^*(a|s) \; \forall \, (s,a) \in \mathcal{S} \times \mathcal{A} \quad &\Rightarrow \quad \prod_{t=0}^{T-1} \pi_{_G}(a_t|s_t) = \prod_{t=0}^{T-1} \tilde{\pi}^*(a_t|s_t) \; \forall \tau \in \mathcal{T} \\ &\Rightarrow \quad q_{\pi_{_G}}(\tau) = q_{\pi_{_E}}(\tau) \; \forall \, \tau \in \mathcal{T} \\ &\Rightarrow \quad D_{\tilde{\pi}^*,\tilde{\pi}^*} = \frac{1}{2} \; \forall \, \tau \in \mathcal{T} \\ &\Rightarrow \quad L(\tilde{\pi}^*, \tilde{\pi}^*) = -\log 4 \end{aligned} \tag{7.12}$$

By multiplying the numerator and denominator of $D_{\tilde{\pi}^*,\tilde{\pi}^*}$ by $\xi(\tau)$ it can be shown in exactly the same way as in the proof of Lemma 1 that $-\log 4$ is the global minimum of $L(\tilde{\pi}^*, \pi_{_G})$. □

Theorem 1's benefits are similar to the ones from Lemma 1: we can use a discriminator of the form of Eq. (7.10) to fit the expert demonstrations with a policy $\tilde{\pi}^*$ that simultaneously yields the optimal generator's policy and produces the same trajectory distribution as the expert policy.

7.1.2 A Specific Policy Class

The derivations of Section 7.1.1 rely on the use of a learnable policy that can both be evaluated and sampled from in order to fit the expert policy. A number of parameterization options that satisfy these conditions are available.

First of all, we observe that since $\pi_{_E}$ is independent of r and π, we can add the entropy of the expert policy $\mathcal{H}(\pi_{_E}(\cdot|s))$ to the MaxEnt IRL objective of Eq. (3.39) without modifying

the solution to the optimization problem:

$$\min_{r\in\mathcal{R}}\left(\max_{\pi\in\Pi} J_\pi[r(s,a)+\mathcal{H}(\pi(\cdot|s))]\right) - J_{\pi_{_E}}[r(s,a)+\mathcal{H}(\pi_{_E}(\cdot|s))] \tag{7.13}$$

The max over policies implies that when optimizing r, π has already been made optimal with respect to the causal entropy augmented reward function $r'(s,a|\pi) = r(s,a)+\mathcal{H}(\pi(\cdot|s))$ and therefore it must be of the form presented in Eq. (3.19). Moreover, since π is optimal w.r.t. r' the difference in performance $J_\pi[r'(s,a|\pi)] - J_{\pi_{_E}}[r'(s,a|\pi_{_E})]$ is always non-negative and its minimum of 0 is only reached when $\pi_{_E}$ is also optimal w.r.t. r', in which case $\pi_{_E}$ must also be of the form of Eq. (3.19).

With discrete action spaces, we propose to parameterize the MaxEnt policy defined in Eq. (3.19) with the following categorical distribution

$$\tilde{\pi}(a|s) = \exp\left(Q_\theta(s,a) - \log\sum_{a'}\exp Q_\theta(s,a')\right),$$

where Q_θ is a model parameterized by θ that approximates $\frac{1}{\alpha}Q^*_{\text{soft}}$.

With continuous action spaces, the soft value function involves an intractable integral over the action domain. Therefore, we approximate the MaxEnt distribution with a Normal distribution with a diagonal covariance matrix like it is commonly done in the literature (Haarnoja et al., 2018; Nachum et al., 2018b). By parameterizing the mean and variance we get a learnable density function that can be easily evaluated and sampled.

7.1.3 Adversarial Soft Advantage Fitting – practical algorithm

Section 7.1.1 shows that assuming $\tilde{\pi}$ can be evaluated and sampled from, we can use the structured discriminator of Eq. (7.10) to learn a policy $\tilde{\pi}$ that matches the expert's trajectory distribution. Section 7.1.2 proposes parameterizations for discrete and continuous action spaces that satisfy those assumptions.

In practice, as with GANs, we do not train the discriminator to convergence as gradient-based optimization cannot be expected to find the global optimum of non-convex problems (Goodfellow et al., 2014). Instead, ASAF alternates between two simple steps:

1. training $D_{\tilde{\pi},\pi_{_G}}$ by minimizing the binary cross-entropy loss,

$$\begin{aligned}
&\mathcal{L}_{BCE}(\mathcal{D}_E,\mathcal{D}_G,\tilde{\pi}) \approx -\frac{1}{n_E}\sum_{i=1}^{n_E}\log D_{\tilde{\pi},\pi_{_G}}(\tau_i^{(E)}) - \frac{1}{n_G}\sum_{i=1}^{n_G}\log\left(1-D_{\tilde{\pi},\pi_{_G}}(\tau_i^{(G)})\right)\\
&\text{where}\quad \tau_i^{(E)}\sim\mathcal{D}_E\ ,\ \tau_i^{(G)}\sim\mathcal{D}_G\\
&\text{and}\ \ D_{\tilde{\pi},\pi_{_G}}(\tau) = \frac{\prod_{t=0}^{T-1}\tilde{\pi}(a_t|s_t)}{\prod_{t=0}^{T-1}\tilde{\pi}(a_t|s_t)+\prod_{t=0}^{T-1}\pi_{_G}(a_t|s_t)}
\end{aligned} \tag{7.14}$$

with minibatch sizes $n_E = n_G$, and
2. updating the generator's policy as $\pi_G \leftarrow \tilde{\pi}$ to minimize Eq. (7.9) (see Algorithm 5).

We derived ASAF considering full trajectories, yet it might be preferable in practice to split full trajectories into smaller chunks. This is particularly true in environments where trajectory length varies a lot or tends to infinity.

To investigate whether the practical benefits of using partial trajectories hurt ASAF's performance, we also consider a variation, ASAF-w, where we treat trajectory-windows of size w as if they were full trajectories. Note that considering windows as full trajectories results in approximating that the initial states of these sub-trajectories have equal probability under the expert's and the generator's policy (this is easily seen when deriving Eq. (7.10)). In the limit, ASAF-1 (window size of 1) becomes a transition-wise algorithm which can be desirable if one wants to collect rollouts asynchronously or has only access to unsequential expert data. While ASAF-1 may work well in practice it essentially assumes that the expert's and the generator's policies have the same state occupancy measure, which is incorrect until actually recovering the true expert policy.

Algorithm 5 ASAF

Require: expert trajectories $\mathcal{D}_E = \{\tau_i\}_{i=1}^{N_E}$
  Randomly initialize $\tilde{\pi}$ and set $\pi_G \leftarrow \tilde{\pi}$
  for steps $m = 0$ to M **do**
    Collect trajectories $\mathcal{D}_G = \{\tau_i\}_{i=1}^{N_G}$ using π_G
    Update $\tilde{\pi}$ by minimizing Eq. (7.14)
    Set $\pi_G \leftarrow \tilde{\pi}$

Finally, to offer a complete family of algorithms based on the structured discriminator approach, we show in the next section that this assumption is not mandatory and derive a transition-wise algorithm based on Soft Q-function Fitting (rather than soft advantages) that also gets rid of the RL loop. We call this algorithm Adversarial Soft Q Fitting (ASQF). While theoretically sound, we found that in practice that ASQF is outperformed by ASAF-1 in more complex environments (see Subsection 7.4.6).

7.2 Transition-wise Imitation Learning without Policy Optimization

In this section, we present Adversarial Soft Q Fitting, a principled approach to Imitation Learning without Reinforcement Learning that relies exclusively on transitions. Using transitions rather than trajectories presents several practical benefits such as the possibility to deal with asynchronously collected data or non-sequential expert demonstrations. We first present the theoretical setting for ASQF and then test it on a variety of discrete control tasks. We show in Subsection 7.4.6 that while it is theoretically sound, ASQF is often outperformed by ASAF-1, an approximation to ASAF that also allows learning on transitions

instead of trajectories.

7.2.1 Adversarial Soft Q Fitting – theoretical setting

We consider the GAN objective of Eq. (3.41) with $x = (s, a)$, $\mathcal{X} = \mathcal{S} \times \mathcal{A}$, $p_{_E} = d_{\pi_{_E}}$, $p_{_G} = d_{\pi_{_G}}$ and a discriminator $D_{\tilde{f},\pi_{_G}}$ of the form of Fu et al. (2017):

$$\begin{aligned} &\min_{\pi_{_G}} \max_{\tilde{f}} L(\tilde{f}, \pi_{_G}) \,, \quad L(\tilde{f}, \pi_{_G}) \triangleq \mathbb{E}_{d_{\pi_{_E}}}[\log D_{\tilde{f},\pi_{_G}}(s,a)] + \mathbb{E}_{d_{\pi_{_G}}}[\log(1 - D_{\tilde{f},\pi_{_G}}(s,a))], \\ &\qquad \text{with} \quad D_{\tilde{f},\pi_{_G}} = \frac{\exp \tilde{f}(s,a)}{\exp \tilde{f}(s,a) + \pi_{_G}(a|s)}, \end{aligned} \tag{7.15}$$

for which we present the following theorem.

Theorem 2. *For any generator policy $\pi_{_G}$, the optimal discriminator parameter for Eq. (7.15) is*

$$\tilde{f}^* \triangleq \arg\max_{\tilde{f}} L(\tilde{f}, \pi_{_G}) = \log \left(\pi_{_E}(a|s) \frac{d_{\pi_{_E}}(s)}{d_{\pi_{_G}}(s)} \right) \; \forall (s,a) \in \mathcal{S} \times \mathcal{A}$$

Using $\tilde{f}^*$, the optimal generator policy $\pi_{_G}^*$ is

$$\arg\min_{\pi_{_G}} \max_{\tilde{f}} L(\tilde{f}, \pi_{_G}) = \arg\min_{\pi_{_G}} L(\tilde{f}^*, \pi_{_G}) = \pi_{_E}(a|s) = \frac{\exp \tilde{f}^*(s,a)}{\sum_{a'} \exp \tilde{f}^*(s,a')} \; \forall (s,a) \in \mathcal{S} \times \mathcal{A}.$$

Proof. The beginning of the proof closely follows the proof of Lemma 1.

$$\begin{aligned} &\arg\max_{\tilde{f}} L(\tilde{f}, \pi_{_G}) = \\ &\qquad \arg\max_{\tilde{f}} \sum_{s_i, a_i} d_{\pi_{_E}}(s_i, a_i) \log D_{\tilde{f},\pi_{_G}}(s_i, a_i) + d_{\pi_{_G}}(s_i, a_i) \log(1 - D_{\tilde{f},\pi_{_G}}(s_i, a_i)) \end{aligned} \tag{7.16}$$

We solve for each individual (s_i, a_i) pair and note that $\tilde{f}_i \mapsto D_i = \dfrac{\exp \tilde{f}_i}{\exp \tilde{f}_i + \pi_{_G,i}}$ is strictly monotonic on $\tilde{f}_i \in \mathbb{R} \, \forall \, \pi_{_G,i} \in]0, 1[$ so,

$$D_i^* = \arg\max_{D_i} L_i \Leftrightarrow \tilde{f}_i^* = \arg\max_{\tilde{f}} L_i \tag{7.17}$$

Taking the derivative and setting it to 0, we find that

$$\left. \frac{dL_i}{d\tilde{f}_i} \right|_{\tilde{f}_i} = 0 \quad \Leftrightarrow \quad \tilde{f}_i = \log \left(\pi_{_G,i} \frac{d_{\pi_{_E},i}}{d_{\pi_{_G},i}} \right) \tag{7.18}$$

We confirm that we have a global maximum with the second derivative test and the values at the border of the domain i.e.,

$$\left.\frac{d^2 L_i}{d\tilde{f}_i^2}\right|_{\tilde{f}_i^*} < 0 \text{ and } L_i \text{ goes to } -\infty \text{ for } \tilde{f}_i \to +\infty \text{ and for } \tilde{f}_i \to -\infty.$$

It follows that

$$\begin{aligned} & \tilde{f}^*(s,a) = \log\left(\pi_{_G}(a|s)\frac{d_{\pi_{_E}}(s,a)}{d_{\pi_{_G}}(s,a)}\right) \quad \forall (s,a) \in \mathcal{S}\times\mathcal{A} \\ \implies & \tilde{f}^*(s,a) = \log\left(\cancel{\pi_{_G}(a|s)}\frac{d_{\pi_{_E}}(s)\pi_{_E}(a|s)}{d_{\pi_{_G}}(s)\cancel{\pi_{_G}(a|s)}}\right) \quad \forall (s,a) \in \mathcal{S}\times\mathcal{A} \\ \implies & \tilde{f}^*(s,a) = \log\left(\pi_{_E}(a|s)\frac{d_{\pi_{_E}}(s)}{d_{\pi_{_G}}(s)}\right) \quad \forall (s,a) \in \mathcal{S}\times\mathcal{A} \end{aligned} \tag{7.19}$$

This proves the first part of Theorem 2.

To prove the second part notice that

$$\begin{aligned} D_{\tilde{f}^*,\pi_{_G}}(s,a) &= \frac{\pi_{_E}(a|s)\frac{d_{\pi_{_E}}(s)}{d_{\pi_{_G}}(s)}}{\pi_{_E}(a|s)\frac{d_{\pi_{_E}}(s)}{d_{\pi_{_G}}(s)} + \pi_{_G}(a|s)} \\ &= \frac{\pi_{_E}(a|s)d_{\pi_{_E}}(s)}{\pi_{_E}(a|s)d_{\pi_{_E}}(s) + \pi_{_G}(a|s)d_{\pi_{_G}}(s)} \\ &= \frac{d_{\pi_{_E}}(s,a)}{d_{\pi_{_E}}(s,a) + d_{\pi_{_G}}(s,a)} \end{aligned} \tag{7.20}$$

This is equal to the optimal discriminator of the GAN objective Eq. (7.5) when $x = (s,a)$. For this discriminator we showed in Lemma 1 that the optimal generator $\pi_{_G}^*$ is such that $d_{\pi_{_G}^*}(s,a) = d_{\pi_{_E}}(s,a)\ \forall (s,a) \in \mathcal{S}\times\mathcal{A}$, which is satisfied for $\pi_{_G}^*(a|s) = \pi_{_E}(a|s)\ \forall (s,a) \in \mathcal{S}\times\mathcal{A}$. Using the fact that

$$\sum_{a'} \exp \tilde{f}^*(s,a') = \sum_{a'} \pi_{_E}(a'|s)\frac{d_{\pi_{_E}}(s)}{d_{\pi_{_G}}(s)} = \frac{d_{\pi_{_E}}(s)}{d_{\pi_{_G}}(s)}\sum_{a'}\pi_{_E}(a'|s) = \frac{d_{\pi_{_E}}(s)}{d_{\pi_{_G}}(s)}. \tag{7.21}$$

we can combine Eq. (7.19) and Eq. (7.21) to write the expert's policy $\pi_{_E}$ as a function of the

optimal discriminator parameter $\tilde{f}^*$:

$$\pi_{_E}(a|s) = \frac{\exp \tilde{f}^*(s,a)}{\sum_{a'} \exp \tilde{f}^*(s,a')} \; \forall (s,a) \in \mathcal{S} \times \mathcal{A}. \tag{7.22}$$

This concludes the second part of the proof. □

7.2.2 Adversarial Soft Q Fitting - practical algorithm

In a nutshell, Theorem 2 tells us that training the discriminator in Eq. (7.15) to distinguish between transitions from the expert and transitions from a generator policy can be seen as retrieving $\tilde{f}^*$ which plays the role of the expert's soft Q-function (i.e., which matches Eq. (3.19) for $\tilde{f}^* = \frac{1}{\alpha} Q^*_{\text{soft},E}$):

$$\pi_{_E}(a|s) = \frac{\exp \tilde{f}^*(s,a)}{\sum_{a'} \exp \tilde{f}^*(s,a')} = \exp\left(\tilde{f}^*(s,a) - \log \sum_{a'} \exp \tilde{f}^*(s,a')\right), \tag{7.23}$$

Therefore, by training the discriminator, one simultaneously retrieves the optimal generator policy.

There is one caveat though: the summation over actions that is required in Eq. (7.23) to go from $\tilde{f}^*$ to the policy is intractable in continuous action spaces and would require an additional step such as a projection to a proper distribution (Haarnoja et al. (2018) use a Gaussian) in order to draw samples and evaluate likelihoods. Updating in this way the generator policy to match a softmax over our learned state-action preferences ($\tilde{f}^*$) becomes very similar in requirements and computational load to a policy optimization step, thus defeating the purpose of this work which is to get rid of the policy optimization step. For this reason, we only consider ASQF for discrete action spaces. As explained in Section 7.1.3, in practice we optimize $D_{\tilde{f},\pi_{_G}}$ only for a few steps before updating $\pi_{_G}$ by normalizing $\exp \tilde{f}(s,a)$ over the action dimension. See Algorithm 6 for the pseudo-code.

Algorithm 6 ASQF

Require: expert transitions $\mathcal{D}_E = \{(s_i, a_i)\}_{i=1}^{N_E}$
Randomly initialize $\tilde{f}$ and get $\pi_{_G}$ from Eq. (7.23)
for steps $m = 0$ to M **do**
  Collect transitions $\mathcal{D}_G = \{(s_i, a_i)\}_{i=1}^{N_G}$ with $\pi_{_G}$
  Train $D_{\tilde{f},\pi_{_G}}$ on $\mathcal{D}_E$ and $\mathcal{D}_G$
  Get $\pi_{_G}$ from Eq. (7.23)

7.3 Experimental setup

We compare our algorithms ASAF, ASAF-w and ASAF-1 against GAIL (Ho and Ermon, 2016), the predominant Adversarial Imitation Learning algorithm in the literature, and AIRL (Fu et al., 2017), one of its variations that also leverages the access to the generator's policy distribution. Additionally, we compare against SQIL (Reddy et al., 2019), a recent Reinforcement Learning-only approach to Imitation Learning that proved successful on high-dimensional tasks. Our implementations of GAIL and AIRL use PPO (Schulman et al., 2017) instead of TRPO (Schulman et al., 2015a) as it has been shown to improve performance (Kostrikov et al., 2019). Finally, we do not use causal entropy regularization to be consistent with (Ho and Ermon, 2016).

For all tasks except MuJoCo, we selected the best-performing hyperparameters through a random search of equal budget for each algorithm-environment pair (see Subsection 7.3.2) and the best configuration is retrained on ten random seeds. For the MuJoCo experiments, GAIL required extensive tuning (through random searches) of both its RL and IRL components to achieve satisfactory performances. Our methods, ASAF-w and ASAF-1, on the other hand, showed much more stability and robustness to hyper parameterization, which is likely due to their simplicity. SQIL used the same implementation and hyperparameters than the SAC (Haarnoja et al., 2018) version that we generated the expert demonstrations from.

Finally, for each task, all algorithms use the same neural network architectures for their policy and/or discriminator (see full description in Subsection 7.3.2). Expert demonstrations are either generated by hand (mountaincar), using open-source bots (Pommerman) or from our implementations of SAC and PPO (all remaining). More details are given in 7.3.1.

7.3.1 Environments and expert demonstrations

Classic Control. The environments used here are the reference Gym implementations for classic control[1] and for Box2D[2]. We generated the expert trajectories for mountaincar (both discrete and continuous versions) by hand using keyboard inputs. For the other tasks, we trained our SAC implementation to get experts on the discrete action tasks and our PPO implementation to get experts on the continuous action tasks. Mountaincar-c and lunarlander-c refer to the continuous action versions of these environments.

MuJoCo. The experts were trained using our implementation of SAC (Haarnoja et al., 2018) the state-of-the-art RL algorithm in MuJoCo continuous control tasks. Our implementation basically refactors the SAC implementation from Rlpyt[3]. We trained the SAC agent for

[1]See: `http://gym.openai.com/envs/#classic_control`
[2]See: `http://gym.openai.com/envs/#box2d`
[3]See: `https://github.com/astooke/rlpyt`

1,000,000 steps for Hopper-v2 and 3,000,000 steps for Walker2d-v2 and HalfCheetah-v2 and Ant-v2. We used the default hyperparameters from Rlpyt.

Pommerman. The observation space that we use for the Pommerman domain (Resnick et al., 2018) is composed of a set of 15 feature maps as well as an additional information vector. The feature maps, whose dimensions are given by the size of the board (8x8 in the case of 1vs1 tasks), are one-hot across the third dimension and represent which element is present at which location. Specifically, these feature maps identify whether a given location is the current player, an ally, an enemy, a passage, a wall, a wood, a bomb, a flame, a fog, or a power-up. Other feature maps contain integers indicating bomb blast strength, bomb life, bomb moving direction, and flame life for each location. Finally, the additional information vector contains the timestep, number of ammunition, whether the player can kick, and its blast strength. The agent has an action space composed of six actions: do-nothing, up, down, left, right, and lay bomb.

For these experiments, we generate the expert demonstrations using Agent47Agent, the open-source champion algorithm of the FFA 2018 competition (Zhou et al., 2018) which uses hardcoded heuristics and Monte Carlo Tree Search[4]. While this agent occasionally eliminates itself during a match, we only select trajectories leading to a win as being expert demonstrations.

Demonstrations summary. Table 7.1 provides a summary of the expert data used.

Task-Name	Expert mean return	Number of expert trajectories
Cartpole	200.0	10
Mountaincar	-108.0	10
Lunarlander	277.5	10
Pendulum	-158.6	10
Mountaincar-c	93.92	10
Lunarlander-c	266.1	10
Hopper	3537	25
Walker2D	5434	25
Halfcheetah	7841	25
Ant	5776	25
Pommerman random-tag	1	300, 150, 75, 15, 5, 1

Table 7.1: Expert demonstrations used for Imitation Learning

[4]See: `https://github.com/YichenGong/Agent47Agent/tree/master/pommerman`

7.3.2 Hyperparameter tuning and best configurations

Classic Control. This first set of experiments uses the fixed hyperparameters of Table 7.2.

RL component

Hyperparameter	Discrete Control	Continuous Control
SAC		
Batch size (in transitions)	256	256
Replay Buffer length $\|\mathcal{B}\|$	10^6	10^6
Warmup (in transitions)	1280	10240
Initial entropy weight α	0.4	0.4
Gradient norm clipping threshold	0.2	1
Transitions between update	40	1
Target network weight τ	0.01	0.01
PPO		
Batch size (in transitions)	256	256
GAE parameter λ	0.95	0.95
Transitions between update	-	2000
Episodes between updates	10	-
Epochs per update	10	10
Update clipping parameter	0.2	0.2

Reward Learning component

Hyperparameter	Discrete Control	Continuous Control
AIRL, GAIL, ASAF-1		
Batch size (in transitions)	256	256
Transitions between update	-	2000
Episodes between updates	10	-
Epochs per update	50	50
Gradient value clipping threshold (ASAF-1)	-	1
ASAF, ASAF-w		
Batch size (in trajectories)	10	10
Episodes between updates	10	20
Epochs per update	50	50
Window size w	(searched)	200
Gradient value clipping threshold	-	1

Table 7.2: Fixed hyperparameters for classic control tasks

For the most sensitive hyperparameters, i.e., the learning rates for the reinforcement learning and discriminator updates (ϵ_{RL} and ϵ_{D}), we perform a random search over 50 configurations and 3 seeds each (for each algorithm on each task) for 500 episodes. We consider logarithmic ranges, i.e., $\epsilon = 10^u$ with $u \sim \text{Uniform}(-6, -1)$ for ϵ_{D} and $u \sim \text{Uniform}(-4, -1)$ for ϵ_{RL}. We also include in this search the critic learning rate coefficient κ for PPO also sampled according to a logarithmic scale with $u \sim Uniform(-2, 2)$ so that the effective learning rate for PPO's critic network is $\kappa \cdot \epsilon_{\text{RL}}$. For discrete action tasks, the window-size w for ASAF-w is sampled uniformly within $\{32, 64, 128\}$. The best configuration for each algorithm is presented in Tables 7.3 to 7.8. Figure 7.1 uses these configurations retrained on 10 seeds and twice as long.

Finally for all neural networks (policies and discriminators) for these experiments we use a fully-connected MLP with two hidden layers and ReLU activation (except for the last layer). We used hidden sizes of 64 for the discrete tasks and 256 for the continuous tasks.

Hyperparameter	ASAF	ASAF-w	ASAF-1	SQIL	AIRL + PPO	GAIL + PPO
Discriminator update lr ϵ_{D}	0.028	0.039	0.00046	-	$2.5*10^{-6}$	0.00036
RL update lr ϵ_{RL}	-	-	-	0.0067	0.0052	0.012
Critic lr coefficient κ	-	-	-	-	0.25	0.29
window size w	-	64	1	-	-	-
window stride	-	64	1	-	-	-

Table 7.3: Best found hyperparameters for Cartpole

Hyperparameter	ASAF	ASAF-w	ASAF-1	SQIL	AIRL + PPO	GAIL + PPO
Discriminator update lr ϵ_{D}	0.059	0.059	0.0088	-	0.0042	0.00016
RL update lr ϵ_{RL}	-	-	-	0.062	0.016	0.0022
Critic lr coefficient κ	-	-	-	-	4.6	0.018
window size w	-	32	1	-	-	-
window stride	-	32	1	-	-	-

Table 7.4: Best found hyperparameters for Mountaincar

hyperparameter	ASAF	ASAF-w	ASAF-1	SQIL	AIRL + PPO	GAIL + PPO
Discriminator update lr ϵ_{D}	0.0055	0.0015	0.00045	-	0.0002	0.00019
RL update lr ϵ_{RL}	-	-	-	0.0036	0.0012	0.0016
Critic lr coefficient κ	-	-	-	-	0.48	8.5
window size w	-	32	1	-	-	-
window stride	-	32	1	-	-	-

Table 7.5: Best found hyperparameters for Lunarlander

HYPERPARAMETER	ASAF	ASAF-w	ASAF-1	SQIL	AIRL + PPO	GAIL + PPO
DISCRIMINATOR UPDATE LR ϵ_{D}	0.00069	0.00082	0.00046	-	$4.3*10^{-6}$	$1.6*10^{-5}$
RL UPDATE LR ϵ_{RL}	-	-	-	0.0001	0.00038	0.00028
CRITIC LR COEFFICIENT κ	-	-	-	-	0.028	84
WINDOW SIZE w	-	200	1	-	-	-
WINDOW STRIDE	-	200	1	-	-	-

Table 7.6: Best found hyperparameters for Pendulum

HYPERPARAMETER	ASAF	ASAF-w	ASAF-1	SQIL	AIRL + PPO	GAIL + PPO
DISCRIMINATOR UPDATE LR ϵ_{D}	0.00021	$3.8*10^{-5}$	$6.2*10^{-6}$	-	$1.7*10^{-5}$	$1.5*10^{-5}$
RL UPDATE LR ϵ_{RL}	-	-	-	0.0079	0.0012	0.0052
CRITIC LR COEFFICIENT κ	-	-	-	-	10	12
WINDOW SIZE w	-	200	1	-	-	-
WINDOW STRIDE	-	200	1	-	-	-

Table 7.7: Best found hyperparameters for Mountaincar-c

HYPERPARAMETER	ASAF	ASAF-w	ASAF-1	SQIL	AIRL + PPO	GAIL + PPO
DISCRIMINATOR UPDATE LR ϵ_{D}	0.0051	0.0022	0.0003	-	0.0045	0.00014
RL UPDATE LR ϵ_{RL}	-	-	-	0.0027	0.00031	0.00049
CRITIC LR COEFFICIENT κ	-	-	-	-	14	0.01
WINDOW SIZE w	-	200	-	-	-	-
WINDOW STRIDE	-	200	-	-	-	-

Table 7.8: Best found hyperparameters for Lunarlander-c

MuJoCo. For MuJoCo experiments (Hopper-v2, Walker2d-v2, HalfCheetah-v2, Ant-v2), the fixed hyperparameters are presented in Table 7.9. For all experiments, fully-connected MLPs with two hidden layers and ReLU activation (except for the last layer) were used, and the number of hidden units is equal to 256.

For SQIL we used SAC with the same hyperparameters used to generate the expert demonstrations. For ASAF, ASAF-1 and ASAF-w, we set the learning rate for the discriminator at 0.001 and ran random searches over 25 randomly sampled configurations and 2 seeds for each task to select the other hyperparameters for the discriminator training. These hyperparameters included the discriminator batch size sampled from a uniform distribution over $\{10, 20, 30\}$ for ASAF and ASAF-w (in trajectories) and over $\{100, 500, 1000, 2000\}$ for ASAF-1 (in transitions), the number of epochs per update sampled from a uniform distribution over $\{10, 20, 50\}$, the gradient norm clipping threshold sampled form a uniform distribution over $\{1, 10\}$, the window-size (for ASAF-w) sampled from a uniform distribution over $\{100, 200, 500, 1000\}$ and the window stride (for ASAF-w) sampled from a uniform distribution over $\{1, 50, w\}$. For GAIL, we obtained poor results using the original hyperparameters from (Ho and Ermon, 2016) for a number of tasks so we ran random searches

RL component	
Hyperparameter	Hopper, Walker2d, HalfCheetah, Ant
PPO (for GAIL)	
GAE parameter λ	0.98
Transitions between updates	2000
Epochs per update	5
Update clipping parameter	0.2
Critic lr coefficient κ	0.25
Discount factor γ	0.99
Reward Learning component	
Hyperparameter	Hopper, Walker2d, HalfCheetah, Ant
GAIL	
Transitions between updates	2000
ASAF	
Episodes between updates	25
ASAF-1 and ASAF-w	
Transitions between updates	2000

Table 7.9: Fixed hyperparameters for MuJoCo environments.

over 100 randomly sampled configurations for each task and 2 seeds to select for the following hyperparameters: the log learning rate of the RL update and the discriminator update separately sampled from uniform distributions over $[-7, -1]$, the gradient norm clipping for the RL update and the discriminator update separately sampled from uniform distributions over $\{None, 1, 10\}$, the number of epochs per update sampled from a uniform distribution over $\{5, 10, 30, 50\}$, the gradient penalty coefficient sampled from a uniform distribution over $\{1, 10\}$ and the batch size for the RL update and discriminator update separately sampled from uniform distributions over $\{100, 200, 500, 1000, 2000\}$.

Hyperparameter	ASAF	ASAF-w	ASAF-1	SQIL	GAIL + PPO
RL batch size (in transitions)	-	-	-	256	200
Discriminator batch size (in transitions)	-	-	100	-	2000
Discriminator batch size (in trajectories)	10	10	-	-	-
Gradient clipping (RL update)	-	-	-	-	1.
Gradient clipping (discriminator update)	10.	10.	1.	-	1.
Epochs per update	50	50	30	-	5
Gradient penalty (discriminator update)	-	-	-	-	1.
RL update lr ϵ_{RL}	-	-	-	$3 * 10^{-4}$	$1.8 * 10^{-5}$
Discriminator update lr ϵ_{D}	0.001	0.001	0.001	-	0.011
window size w	-	200	1	-	-
window stride	-	1	1	-	-

Table 7.10: Best found hyperparameters for the Hopper-v2 environment

Hyperparameter	ASAF	ASAF-w	ASAF-1	SQIL	GAIL + PPO
RL batch size (in transitions)	-	-	-	256	1000
Discriminator batch size (in transitions)	-	-	100	-	100
Discriminator batch size (in trajectories)	10	10	-	-	-
Gradient clipping (RL update)	-	-	-	-	-
Gradient clipping (discriminator update)	10.	1	1	-	10
Epochs per update	50	10	30	-	30
Gradient penalty (discriminator update)	-	-	-	-	1.
RL update lr ϵ_{RL}	-	-	-	$3 * 10^{-4}$	0.0006
Discriminator update lr ϵ_{D}	0.001	0.001	0.001	-	0.023
window size w	-	200	1	-	-
window stride	-	1	1	-	-

Table 7.11: Best found hyperparameters for the HalfCheetah-v2 environment

Hyperparameter	ASAF	ASAF-w	ASAF-1	SQIL	GAIL + PPO
RL batch size (in transitions)	-	-	-	256	200
Discriminator batch size (in transitions)	-	-	500	-	2000
Discriminator batch size (in trajectories)	20	20	-	-	-
Gradient clipping (RL update)	-	-	-	-	-
Gradient clipping (discriminator update)	10.	1.	10.	-	-
Epochs per update	30	10	50	-	30
Gradient penalty (discriminator update)	-	-	-	-	1.
RL update lr ϵ_{RL}	-	-	-	$3 * 10^{-4}$	0.00039
Discriminator update lr ϵ_{D}	0.001	0.001	0.001	-	0.00066
window size w	-	100	1	-	-
window stride	-	1	1	-	-

Table 7.12: Best found hyperparameters for the Walker2d-v2 environment

Hyperparameter	ASAF	ASAF-w	ASAF-1	SQIL	GAIL + PPO
RL batch size (in transitions)	-	-	-	256	500
Discriminator batch size (in transitions)	-	-	100	-	100
Discriminator batch size (in trajectories)	20	20	-	-	-
Gradient clipping (RL update)	-	-	-	-	-
Gradient clipping (discriminator update)	10.	1.	1.	-	10.
Epochs per update	50	50	10	-	50
Gradient penalty (discriminator update)	-	-	-	-	10
RL update lr ϵ_{RL}	-	-	-	$3 * 10^{-4}$	$8.5 * 10^{-5}$
Discriminator update lr ϵ_{D}	0.001	0.001	0.001	-	0.0016
window size w	-	200	1	-	-
window stride	-	50	1	-	-

Table 7.13: Best found hyperparameters for the Ant-v2 environment

Pommerman. For this set of experiments, we use a number of fixed hyperparameters for all algorithms either inspired from their original papers for the baselines or selected through preliminary searches. These fixed hyperparameters are presented in Table 7.15.

For the most sensitive hyperparameters, the learning rates for the reinforcement learning and discriminator updates (ϵ_{RL} and ϵ_{D}), we perform a random search over 25 configurations and 2 seeds each for all algorithms. We consider logarithmic ranges, i.e., $\epsilon = 10^u$ with $u \sim \text{Uniform}(-7, -3)$ for ϵ_{D} and $u \sim \text{Uniform}(-4, -1)$ for ϵ_{RL}. We also include in this search the window-size w for ASAF-w, sampled uniformly within $\{32, 64, 128\}$. The best configuration for each algorithm is presented in Table 7.14. Figure 7.3 uses these configurations retrained on 10 seeds.

Finally, for all neural networks (policies and discriminators) we use the same architecture. Specifically, we first process the feature maps (see Section 7.3.1) using a 3-layers convolutional network with a number of hidden feature maps of 16, 32, and 64 respectively. Each one of these layers uses a kernel size of 3x3 with a stride of 1, no padding, and a ReLU activation. This module ends with a fully connected layer of hidden size 64 followed by a ReLU activation. The output vector is then concatenated to the unprocessed additional information vector (see Section 7.3.1) and passed through a final MLP with two hidden layers of size 64 and ReLU activations (except for the last layer).

Hyperparameter	ASAF	ASAF-w	ASAF-1	SQIL	AIRL + PPO	GAIL + PPO	BC
Discriminator update lr ϵ_{D}	0.0007	0.0002	0.0001	-	$3.1*10^{-7}$	$9.3*10^{-7}$	0.00022
RL update lr ϵ_{RL}	-	-	-	0.00019	0.00017	0.00015	-
window size w	-	32	1	-	-	-	-
window stride	-	32	1	-	-	-	-

Table 7.14: Best found hyperparameters for the Pommerman Random-Tag environment

RL component	
Hyperparameter	Pommerman Random-Tag
SAC	
Batch size (in transitions)	256
Replay Buffer length $\|\mathcal{B}\|$	10^5
Warmup (in transitions)	1280
Initial entropy weight α	0.4
Gradient norm clipping threshold	0.2
Transitions between update	10
Target network weight τ	0.05
PPO	
Batch size (in transitions)	256
GAE parameter λ	0.95
Episodes between updates	10
Epochs per update	10
Update clipping parameter	0.2
Critic lr coefficient κ	0.5

Reward Learning component	
Hyperparameter	Pommerman Random-Tag
AIRL, GAIL, ASAF-1	
Batch size (in transitions)	256
Episodes between updates	10
Epochs per update	10
ASAF, ASAF-w	
Batch size (in trajectories)	5
Episodes between updates	10
Epochs per update	10

Table 7.15: Fixed Hyperparameters for Pommerman Random-Tag environment.

7.4 Results

We evaluate our methods on a variety of discrete and continuous control tasks. Our results show that, in addition to drastically simplifying the adversarial IRL framework, our methods

perform on par or better than previous approaches in all but one environment. When trajectory length is really long or drastically varies across episodes (see MuJoCo experiments Section 7.4.2), we find that using sub-trajectories with fixed window-size (ASAF-w or ASAF-1) significantly outperforms its full trajectory counterpart ASAF.

7.4.1 Experiments on classic control and Box2D tasks (discrete and continuous)

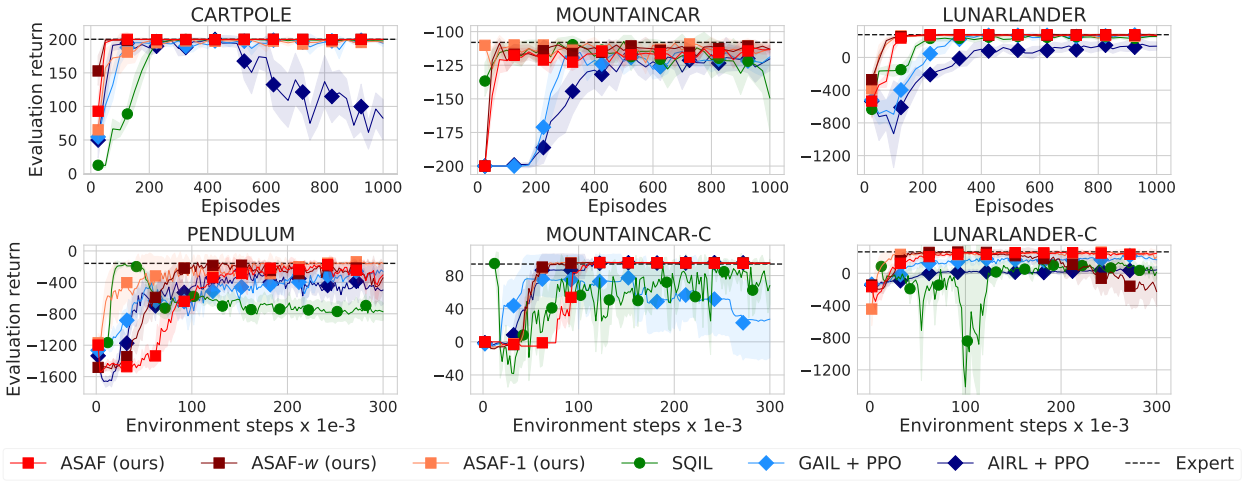


Figure 7.1: Results on classic control and Box2D tasks for 10 expert demonstrations. The first row contains discrete actions environments, the second row corresponds to continuous control.

Figure 7.1 shows that ASAF and its approximate variations ASAF-1 and ASAF-w quickly converge to expert's performance (here w was tuned to values between 32 to 200, see Subsection 7.3.2 for selected window-sizes). This indicates that the practical benefits of using shorter trajectories or even just transitions do not hinder performance on these simple tasks. Note that for Box2D and classic control environments, we retrain the best configuration of each algorithm for twice as long than was done in the hyperparameter search, which allows for uncovering unstable learning behaviors. Figure 7.1 shows that our methods display much more stable learning: their performance rises until they match the expert's and does not decrease once it is reached. This is a highly desirable property for an Imitation Learning algorithm since in practice one does not have access to a reward function and thus cannot monitor the performance of the learning algorithm to trigger early stopping. The baselines on the other hand experience occasional performance drops. For GAIL and AIRL, this is likely due to the concurrent RL and IRL loops, whereas, for SQIL, it has been noted that an effective reward decay can occur when accurately mimicking the expert (Reddy et al.,

2019). This instability is particularly severe in the continuous control case. In practice, all three baselines use early stopping to avoid performance decay (Reddy et al., 2019).

7.4.2 Experiments on MuJoCo (continuous control)

To scale up our evaluations in continuous control we use the popular MuJoCo benchmarks. In this domain, the trajectory length is either fixed at a large value (1000 steps on HalfCheetah) or varies a lot across episodes due to termination when the character falls down (Hopper, Walker2d, and Ant). Figure 7.2 shows that these trajectory characteristics hinder ASAF's learning as this latter requires collecting multiple episodes for every update, while ASAF-1 and ASAF-w perform well and are more sample-efficient than ASAF in these scenarios. We focus on GAIL since Fu et al. (2017) claims that AIRL performs on par with it on MuJoCo environments. In Figure 7.7 in Subsection 7.4.7 we evaluate GAIL both with and without Gradient Penalty (GP) on discriminator updates (Gulrajani et al., 2017; Kostrikov et al., 2019) and while GAIL was originally proposed without GP (Ho and Ermon, 2016), we empirically found that GP prevents the discriminator to overfit and enables RL to exploit dense rewards, which highly improves its sample efficiency. Despite these ameliorations, GAIL proved to be quite inconsistent across environments despite substantial efforts on hyperparameter tuning. On the other hand, ASAF-1 performs well across all environments. Finally, we see that SQIL's instability is exacerbated on MuJoCo.

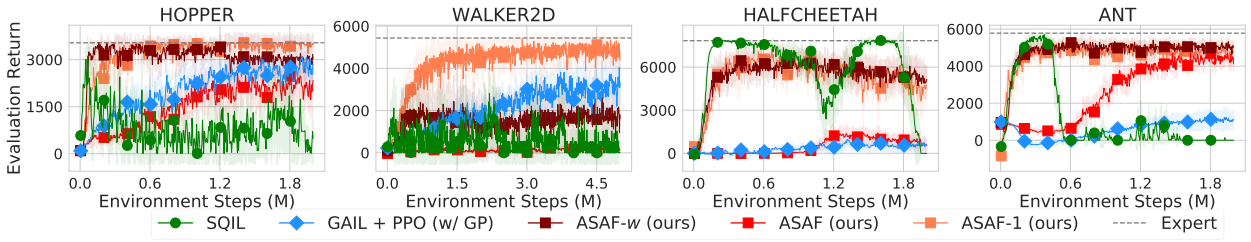


Figure 7.2: Results on MuJoCo tasks for 25 expert demonstrations.

7.4.3 Experiments on Pommerman (discrete control)

Finally, to scale up our evaluations in discrete control environments, we consider the domain of Pommerman (Resnick et al., 2018), a challenging and very dynamic discrete control environment that uses rich and high-dimensional observation spaces (see Subsection 7.3.1). We perform evaluations of all of our methods and baselines on a 1 vs 1 task where a learning agent plays against a random agent, the opponent. The goal for the learning agent is to navigate to the opponent and eliminate it using expert demonstrations provided by the champion algorithm of the FFA 2018 competition (Zhou et al., 2018). We removed the ability of the opponent to lay bombs so that it does not accidentally eliminate itself. Since it can still move

around, it is however surprisingly tricky to eliminate: the expert has to navigate across the whole map, lay a bomb next to the opponent, and retreat to avoid eliminating itself. This entire routine has then to be repeated several times until finally succeeding since the opponent will often avoid the hit by chance. We refer to this task as *Pommerman Random-Tag*. Note that since we measure the success of the imitation task with the win-tie-lose outcome (a notably sparse performance metric), a learning agent has to truly reproduce the expert behavior until the very end of trajectories to achieve higher scores. Figure 7.3 shows that all three variations of ASAF as well as Behavioral Cloning outperform the other baselines.

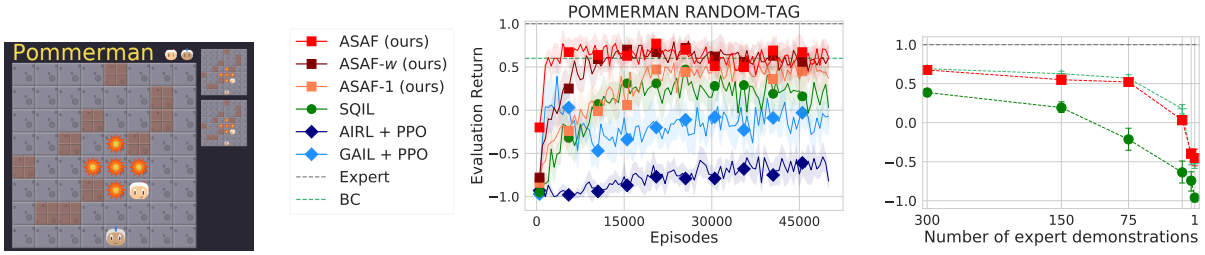


Figure 7.3: Results on Pommerman Random-Tag: (Left) Snapshot of the environment. (Center) Learning measured as evaluation return over episodes for 150 expert trajectories (Right) Average return on last 20% of training for decreasing number of expert trajectories [300, 150, 75, 15, 5, 1].

7.4.4 Wall Clock Time

We report training times in Figure 7.4 and observe that ASAF-1 is always the fastest to learn. Note however that reports of performance w.r.t wall-clock time should always be taken with a grain of salt as they are greatly influenced by hyperparameters and implementation details.

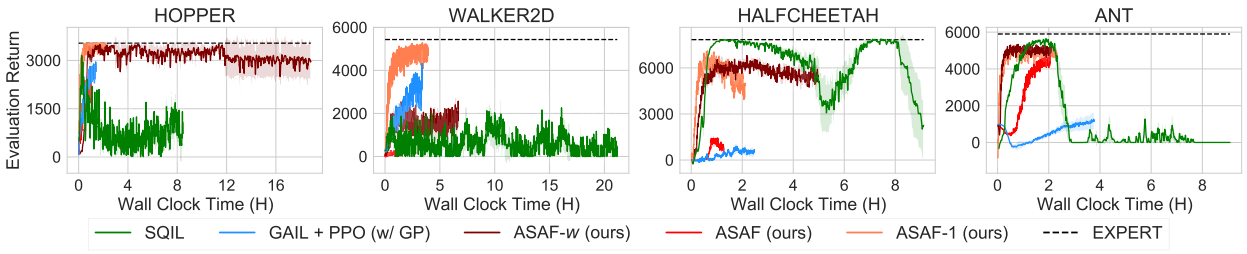


Figure 7.4: Training times on MuJoCo tasks for 25 expert demonstrations.

7.4.5 Mimicking the expert

To ensure that our method actually mimics the expert and does not just learn a policy that collects high rewards when trained with expert demonstrations, we ran ASAF-1 on the Ant-v2 MuJoCo environment using various sets of 25 demonstrations. These demonstrations were generated from a SAC agent at various levels of performance during its training. Since at low levels of performance, the variance of the episode's return is high, we filtered collected demonstrations to lie in the targeted range of performance (e.g. return in [800, 1200] for the 1K set). Results in Figure 7.5 show that our algorithm succeeds at learning a policy that closely emulates various demonstrators (even when these are suboptimal).

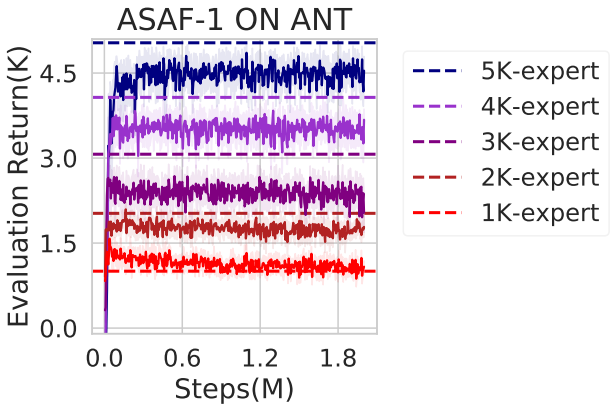


Figure 7.5: ASAF-1 on Ant-v2. Colors are 1K, 2K, 3K, 4K, and 5K expert's performance.

7.4.6 ASQF vs ASAF

Figure 7.6 shows that ASQF performs well in small-scale environments but struggles and eventually fails in more complicated environments. Specifically, it seems that ASQF does not scale well with the observation space size. Indeed mountaincar, cartpole, lunarlander, and pommerman have respectively an observation space dimensionality of 2, 4, 8, and 960. This may be due to the fact that the partition function Eq. (7.21) becomes more difficult to learn. Indeed, for each state, several transitions with different actions are required in order to learn it. Poorly approximating this partition function could lead to assigning too low a probability to expert-like actions and eventually failing to behave appropriately. ASAF on the other hand explicitly learns the probability of an action given the state – in other words, it

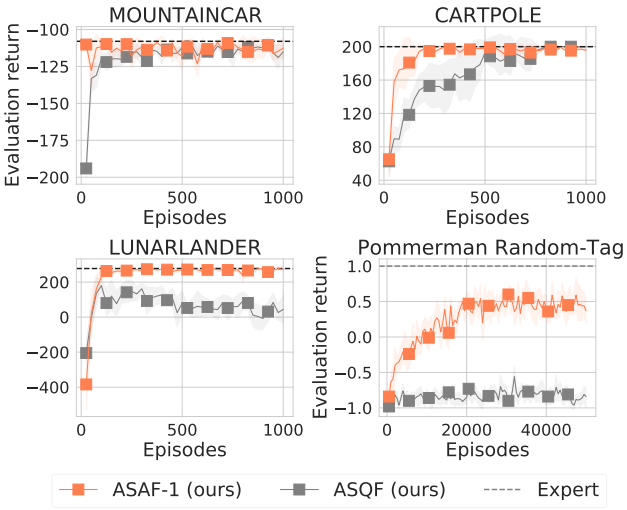


Figure 7.6: Comparison between ASAF-1 and ASQF, our two transition-wise methods, on environments with increasing observation space dimensionality

explicitly learns the partition function – and is therefore immune to that problem.

7.4.7 Importance of Gradient Penalty for GAIL

Figure 7.7 shows the benefits of using Gradient Penalty with GAIL on MuJoCo tasks.

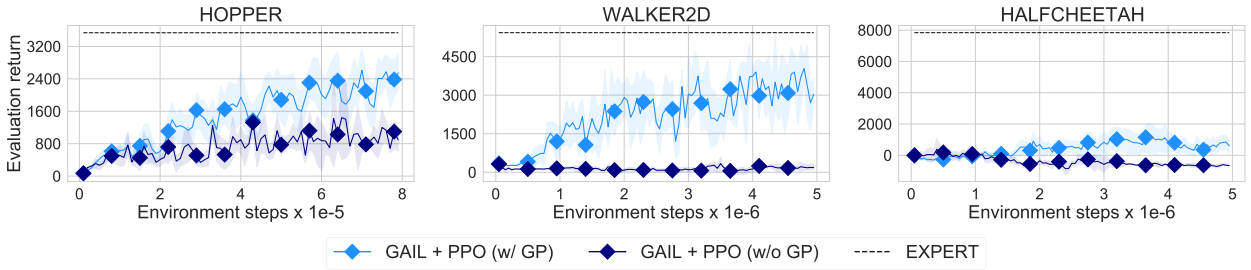


Figure 7.7: Comparison between original GAIL (Ho and Ermon, 2016) and GAIL with Gradient Penalty (Gulrajani et al., 2017; Kostrikov et al., 2019)

7.5 Discussion

MaxEnt IRL, the foundation of modern IL was first proposed in Ziebart et al. (2008). Then, Ziebart (2010) further elaborated it and derived the optimal form of the MaxEnt policy which is at the core of our methods. Finn et al. (2016a) proposed a GAN formulation to IRL that leverages the energy-based models from Ziebart (2010). However, Finn et al. (2016b)'s implementation of this method relied on processing full trajectories with a Linear Quadratic Regulator, and optimizing with guided policy search was required to manage the high variance from the trajectories. To retrieve robust rewards, Fu et al. (2017) proposed a straightforward transposition of Finn et al. (2016a) to state-action transitions. Yet, to do so, they had to let go of the GAN objective (i.e., the Jensen-Shannon divergence) for the policy optimization, and instead minimize the Kullback–Leibler divergence from the expert occupancy measure to the policy occupancy measure (Ghasemipour et al., 2019).

Later works move away from the Generative Adversarial formulation (Sasaki et al., 2018; Kostrikov et al., 2020). To do so, Sasaki et al. (2018) directly express the expectation of the Jensen-Shannon divergence between the occupancy measures in term of the agent's Q-function, which can then be used to optimize the agent's policy with off-policy Actor-Critic (Degris et al., 2012). Similarly, Kostrikov et al. (2020) use Dual Stationary Distribution Correction Estimation (Nachum et al., 2019) to approximate the Q-function on the expert's demonstrations before optimizing the agent's policy under the initial state distribution using the reparametrization trick (Haarnoja et al., 2018). While Sasaki et al. (2018); Kostrikov et al. (2020) are related to our methods in their interests in learning directly the value function, they differ in their goal and thus in the resulting algorithmic complexity. Indeed,

they aim at improving the sample efficiency in terms of environment interaction and therefore move away from the GAN formulation towards more complicated divergence minimization methods. In doing so, they further complicate the Imitation Learning methods while still requiring explicitly learning a policy. Additionally, simply using the GAN formulation with an Experience Replay Buffer can significantly improve sample efficiency (Kostrikov et al., 2019). For these reasons, and since our aim is to propose efficient yet simple methods, we focus on the Adversarial Imitation Learning formulation and the MaxEnt IRL framework.

While Reddy et al. (2019) share our interest in simpler IL methods, they pursue an opposite approach to ours. They propose to eliminate the reward learning steps of IRL by simply hard-coding a reward of 1 for the expert's transitions and of 0 for the agent's transitions. They then use Soft Q-learning (Haarnoja et al., 2017) to learn a value function by sampling transitions in equal proportion from the expert's and agent's buffers. Unfortunately, once the learner accurately mimics the expert, it collects expert-like transitions that are labeled with a reward of 0 since they are generated and not coming from the demonstrations. This effectively causes the reward of expert-like behavior to decay as the agent improves and can severely destabilize learning to a point where early stopping becomes required (Reddy et al., 2019).

Our work builds on Finn et al. (2016a), yet its novelty is to explicitly express the probability of a trajectory in terms of the policy in order to directly learn this latter when training the discriminator. In contrast, Fu et al. (2017) considers a transition-wise discriminator with un-normalized probabilities which makes it closer to ASQF (Section 7.2) than to ASAF-1. Additionally, AIRL from Fu et al. (2017) minimizes the KL divergence between occupancy measures (Ghasemipour et al., 2019) whereas ASAF minimizes the JS divergence between trajectory distributions.

Finally, Behavioral Cloning uses the loss function from supervised learning (classification or regression) to match the expert's actions given the expert's states. Its data is limited to the demonstrated state-action pairs without environment interaction and therefore it suffers from the compounding error problem due to co-variate shift (Ross and Bagnell, 2010). Contrarily, ASAF-1 uses the binary cross entropy loss in Eq. (7.14) and does not suffer from compounding errors as it learns on both generated and expert's trajectories.

Concretely, we strongly believe that enabling affordable agent modeling by developing simple and robust, yet efficient, IL algorithms can pave the way to significant improvements in multi-agent learning. To that end, we propose an important simplification to the Adversarial Imitation Learning framework by removing the Reinforcement Learning optimization loop altogether. We show that, by using a particular form for the discriminator, our method recovers a policy that matches the expert's trajectory distribution. We evaluate our approach against prior works on many different benchmarking tasks and show that our method (ASAF) compares favorably to the predominant Imitation Learning algorithms. The approximate versions, ASAF-w and ASAF-1, that use sub-trajectories yield flexible

algorithms that work well both on short and long-time horizons while remaining simple, robust, and computationally affordable. Finally, our approach still involves a reward learning module through its discriminator, and it would be interesting in future work to explore how ASAF can be used to learn robust rewards, along the lines of Fu et al. (2017).

Chapter 8

Discussion of findings

In this thesis, we highlight both the importance and challenges of coordinating concurrent learners in cooperative multi-agent tasks. Specifically, we strive to identify settings in which multi-agent coordination is strenuous, yet a requirement for good team performance. For each of these settings, we identify the causes for the failure of coordination and propose practical algorithmic improvements that promote coordination and lead to successful teams.

We started with Multi-Agent Reinforcement Learning, the most favorable setting for coordination: all the agents receive rewards and they continuously interact in the environment. Therefore, each agent is driven by explicit learning signals, and learning coordinated behavior through unrestricted interactions with the world or other agents is straightforward.

Nevertheless, we observe that, even in these ideal conditions, prominent learning algorithms fail to coordinate and do not discover the optimal team strategies. Indeed, current MARL algorithms may fall short in training agents that leverage information about the behavior of their teammates. Crucially, this occurs even when agents are explicitly given their teammates' observations, actions, and current policies throughout training. We believe that this is an important finding worth raising some concerns among the community. Particularly, there is a widespread belief that centralized training (like MADDPG) solves coordination and thus should always outperform decentralized training (like DDPG). Not only is this belief unsupported by empirical evidence (at least in our experiments), but it also prevents the community from investigating and tackling coordination flaws. These are important limitations that impede learning safer and more effective multi-agent behaviors. An agent cannot adapt to a new teammate or a change in an ally's behavior if it does not account for others. This prevents current methods to be applied in real-world settings where external perturbations and uncertainties force artificial agents to be flexible and adaptable when interacting with various individuals.

We propose to focus on coordination and give a practical definition of it: an agent's behavior should be predictable given its teammates' behaviors. While we agree that this

definition is restrictive and incomplete, we believe that it is a good starting point to consider. Indeed, enforcing that criterion should make learning agents more aware of their teammates if they are to coordinate with them. Yet, coordination alone does not ensure success, as agents could be coordinated in an unproductive manner. More so, coordination could have detrimental effects if it enables an attacker to influence an agent by taking control of a teammate or using a mock-up teammate. For these reasons, when using MARL algorithms (or even single-agent RL for that matter) for real-world applications, additional safeguards are absolutely required to prevent the system from misbehaving, which is highly probable if out-of-distribution states are to be encountered.

We show that coordination can be promoted by relying on shared incentives and enforcing social norms across agents. First, we motivate the use of coordinated policies to ease the discovery of successful team strategies in cooperative multi-agent tasks. Then, we foster coordination with auxiliary tasks that bias the agent's learning towards desirable policies. We investigate two distinct approaches in Centralized Training and Decentralized Execution MARL algorithms. The first one, TeamReg, regularizes the agents' policies so that agents are both predictable and able to predict their teammates' behaviors. The second one, CoachReg, enforces that agents use sub-policies and that the group switches between different behaviors synchronously and coherently. While the benefits of TeamReg appear task-dependent – we show for example that it can be detrimental on tasks with a competitive component – CoachReg significantly improves performance in almost all the presented coordination-intensive multi-agent problems. Finally, we analyze the effects of our methods on the policies that our agents learn. We observe that the auxiliary tasks successfully enforce the features that we propose as proxies for coordinated behaviors, namely predictability, synchronicity, and coherence. Motivated by the success of this single-step coordination technique, a promising direction is to explore model-based planning approaches to promote coordination over long-term multi-agent interactions.

After exploring online MARL, the most favorable setting, we explore coordination in the more challenging offline setting. An offline RL agent cannot interact with its environment to figure out how its behavior is going to pan out. In offline MARL, the situation is even direr as agents cannot interact with one another to understand how individual strategies will blend together and result in complex team behaviors. Worse, offline learners may have no way to probe each other and gain information about other agents' current policies. This gives rise to what we call the Offline Coordination Problem, which we propose decomposes into the Strategy Agreement and the Strategy Fine-Tuning challenges.

We show that current state-of-the-art offline MARL methods fail at Strategy Agreement and Strategy Fine-Tuning, and so even when allowed to fine-tune online once the offline training is over. Specifically, independent learner approaches fail at both Strategy Agreement and Strategy Fine-Tuning while Centralized Training and Decentralized Execution approaches are able to deal with Strategy Agreement. Surprisingly, no model-free method

is able to tackle Strategy Agreement, even the fully centralized one. The more different the learned agents' behavior from the offline data, the more severe the Strategy Fine-Tuning requirement and thus the more severe the performance drop. This occurs for instance with sub-optimal offline data (agents will try to improve on the collected data and their policy is therefore likely to change) or if partial observability requires agents to adapt their behavior compared to the demonstrated one.

We propose to remediate this Offline Coordination Problem by leveraging centralized training and model-based methods. We learn a world model from the offline data and use it to generate synthetic interaction data. That way, the offline learners can simulate how their individual policies will interact and collectively derive coordinated team strategies. We show that our approach, MOMA-PPO, the first model-based offline MARL method, largely outperforms the prominent offline learning algorithms. Even more impressively, this multi-agent algorithm outperforms single-agent approaches that completely sidestep the Strategy Agreement (SA) problem. This highlights that our method solves both the Strategy Agreement and Strategy Fine-Tuning problems. This suggests that the benefits of model-based approaches observed in the single-agent offline setting (Yu et al., 2020) transfer well to multi-agent problems and can overcome the challenges of offline coordination.

A promising extension to this work is to consider decentralized training. In real-world applications, many offline learners are independent (humans, animals, robots from different companies, etc.) and do not have direct access to other agents' policies. Therefore, effort must be made to propose more relevant learning settings in which offline learners might only be able to communicate with one another during training. Even more restricting settings without communication would require learners to estimate other agents' policies solely from what might be learned from the shared offline data. Finally, research on how to safely learn and leverage world models when querying them outside of the training distribution is mandatory for real-world applications.

After considering MARL and offline MARL, we challenge coordination in a less conventional learning paradigm. While Chapter 5 is about removing "interactions" and questioning if coordination can occur without them, Chapter 6 questions two other pillars of coordination: shared incentives and explicit external learning signals. In classical multi-agent learning paradigms, agents' learning and coordination are guided by either external rewards or demonstrations. In Chapter 6, we take inspiration from the field of experimental semiotics – which has shown the extent of human proficiency at learning from a priori unknown instructions meanings in the absence of clear external reinforcement or demonstrations – and question this assumption. We present the Architect-Builder Problem: an asymmetrical setting in which an architect must learn to guide a builder toward completing a specific task. The architect knows the target task but cannot act in the environment and can only send arbitrary messages to the builder. The builder on the other hand can act in the environment, but receives no rewards nor has any knowledge about the task: it must learn to solve

it by relying only on the messages sent by the architect. Crucially, the meaning of messages is not defined initially, nor shared between the agents, but must be negotiated throughout learning. Under these constraints, we propose Architect-Builder Iterated Guiding, a solution to the ABP where the architect leverages a learned model of the builder to guide it. The builder in return uses self-imitation learning to reinforce the guided behavior. To palliate the non-stationarity induced by the two agents concurrently learning, ABIG structures the sequence of interactions between the agents into interaction frames. We analyze the key learning mechanisms of ABIG and test it in a 2-dimensional instantiation of the ABP where tasks involve grasping cubes, placing them at a given location, or building various shapes. In this environment, ABIG results in a low-level, high-frequency, guiding communication protocol that, not only enables an architect-builder pair to solve the task at hand but can also generalize to unseen tasks.

This work investigates an original approach to autonomous agent learning and coordination by proposing a novel interactive learning setting. The proposed learning paradigm contrasts with more classical supervision such as reward signals and demonstrations. By proposing an iterative and interactive learning framework, the ABP promotes a finer and more flexible control over the learned behaviors than designing rewards or demonstrations. Indeed, the behavior is constantly evaluated and refined throughout learning as the interactions unfold. Still, in this process, it is essential to keep in mind the importance of the architect since it is the agent that judges whether or not the learned behavior is satisfactory.

Our approach has several limitations which open up different opportunities for further work. First, ABIG trains agents in a stationary configuration which implies doing several interaction frames. Each interaction frame involves collecting numerous transitions and thus ABIG is not data efficient. A challenging avenue would be to relax this stationarity constraint and have agents learn from experience buffers containing non-stationary data with obsolete agent behaviors. Second, the builder remains dependent on the architect's messages even at convergence. Using a Vygotskian approach (Colas et al., 2020, 2021), the builder could internalize the guidance from the architect and become autonomous at the task. This could, for instance, be achieved by having the builder learn a model of the architect's message policy once the communication protocol has converged.

Because we present the first step towards interactive agents that learn in the ABP, our method uses simple tools (feed-forward networks and self-imitation learning). It is however important to note that our proposed formulation of the ABP can support many different research directions. Experimenting with agents' models could allow for the investigation of other forms of communication. One could, for instance, include memory mechanisms in the models of agents in order to facilitate the emergence of retrospective feedback, a form of emergent communication observed in Vollmer et al. (2014). ABP is also compatible with low-frequency feedback and to experiment in this direction, one could penalize the architect for sending messages and assess whether a pair can converge to higher-level meanings. Messages could also be composed of several tokens in order to allow for the emergence of

compositionality. Finally, while we believe that the proposed framework is best suited to model Brain-Computer Interface, it can also serve as a testbed to study the fundamental mechanisms of emergent communication and investigate the impact of high-level communication priors of experimental semiotics.

Our findings align with the main axes that we have identified in the literature and discussed in Chapter 2: interactions, internal models and shared incentives are central to coordination. Coordination is about structuring interactions at evaluation and, in turn, requires to structure interactions during learning. In Chapter 4, shaping learning interactions with social norms that enforce predictability, coherence, and synchronicity resulted in more coordinated and better-performing teams. Similarly, ABIG in Chapter 6 relies on *interaction frames* to interleave agents' interactive learning and enable them to coordinate and communicate in a challenging setting. However, it is the Offline Coordination Problem of Chapter 5 that illustrates best the need for interactions: in the absence of them, offline learners cannot coordinate and fail. Yet, leveraging synthetic data to simulate interactions we were able to restore the learners' ability to coordinate and perform together.

Shared incentives are usually what drive coordination, and MARL relies on every agent receiving an environmental reward to foster cooperation. Most of the time, as in Chapters 4 and 5, cooperative MARL these rewards are shared and account for the team performance. On the one hand, we have shown in Chapter 4 that additional shared incentives such as social norms can greatly improve coordination and performance. On the other hand, as we have investigated in Chapter 6, removing this shared team reward assumption makes learning, let alone coordinating, daunting. Learning in this setting required to impose a *shared intent* prior by specifying that agents assume that they pursue the same objective as their teammates. Because some agents can ignore what the objective actually is, this shared incentive is more versatile and can accommodate more settings than shared team rewards. It remains, however, similar to it in essence.

Throughout this thesis, the importance of modeling others and their interactions has become evident. Indeed, it is at the core of all the solutions that we proposed: in Chapter 4, models of others are used to enforce social norms on the leaned agents' policies; in Chapter 5, learning a world model enables us to simulate the learners' interactions and estimate how individual policies will blend into group behaviors. This shows that world models shall not be discarded for multi-agent problems, finally, in Chapter 6, the architect must learn a model of the builder in order to plan how to best guide it. However, current approaches to building inner models of others are either simplistic – lacking expressivity – or overly complex. In practice, this complexity prevents sophisticated methods to be used as part of higher-level multi-agent algorithms. Consequently, in Chapter 7, we propose Adversarial Soft Advantage Fitting, a novel IL approach that is competitive with the state-of-the-art methods while being much more simple, robust, and computationally affordable.

Chapter 9

Conclusion and summary

Our research has highlighted that coordination is not a given and can elude us even in settings that meet all the necessary prerequisites (see Chapter 4). Yet, our investigation has also demonstrated the remarkable versatility of multi-agent learning frameworks and how they allow for efficient coordination when approached appropriately. Notably, we have showcased that coordination can be enhanced by leveraging shared incentives and social norms (see Chapter 4). Moreover, our findings indicate that coordination can occur even in the absence of genuine interactions provided that learners construct world models to simulate how individual behaviors blend into coherent group strategies (see Chapter 5). Remarkably, learners equipped with high-level priors on objectives and interactions can effectively coordinate without relying on external rewards or demonstrations (see Chapter 6). Lastly, Chapter 7 has introduced simple yet efficient Imitation Learning approaches to build models of others, which is a central aspect of coordination.

Our research has delved into diverse settings, revealing that coordination can be attained through the use of interactions, shared incentives, and inner models. However, significant challenges remain before these techniques can alleviate complex and long-term real-world issues. To keep this grounded, let us consider the detrimental impact of human activities and climate change on biodiversity which is well illustrated by the pressing need for safeguarding the integrity of *blue corridors* (Dunn et al., 2019; Johnson et al., 2022). Blue corridors are whales' crucial migratory routes across the oceans and have become very hazardous due to the cumulative threats of industrial fishing, ship strikes, pollution, habitat loss, and climate change. It is complex yet vital to protect these migration superhighways by creating networks of marine protected areas to ensure that whale populations have every opportunity to thrive.

In this context, we believe that leveraging agent-based modeling, mechanism design, and multi-agent learning holds significant potential to inform decision-making processes. For instance, by modeling how climate change will impact oceans' temperatures, salinity, and currents, we can estimate the potential effects on plankton populations and their distribution

across the oceans. Subsequently, this knowledge can inform us about whales' future migratory routes, therefore guiding the design and enforcement of adequate marine protected areas. However, it is crucial to recognize that such interventions may have ripple effects on fishing companies and maritime transport, prompting them to alter their operational strategies, and retroactively affecting whales in unforeseen ways.

This complex scenario is a highly dynamic and intricate multi-agent system for which current techniques fall short. Firstly, the agents are immersed in an environment that constantly evolves, necessitating the integration of sophisticated prediction models from climatology, fluid dynamics, ecology, and economics. Additionally, the multitude of agents involved in this system exhibit heterogeneity and engage in both competitive and cooperative relationships. Moreover, these agents encompass entities such as plankton, whales, fishing, and maritime transportation companies which adapt at very different time scales.

Addressing these challenges demands further advancements in our understanding of multi-agent systems and how they can decompose problems whose complexity emerges from the interaction of many different actors. By bridging the gaps between disciplines to draw insights from agent-based modeling, mechanism design, and multi-agent learning, we can aspire to capture the intricacies that link the agents' behaviors to their environment. Modeling these relationships will eventually allow the creation of comprehensive frameworks to assist in navigating the complexities of real-world decision-making. Ultimately, our research investigates the mechanisms of multi-agent coordination and contributes to the advancement of knowledge that will pave the way for more effective strategies.

Bibliography

Pieter Abbeel and Andrew Y Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the 21st International Conference on Machine Learning (ICML)*, 2004.

David H Ackley and Michael L Littman. Altruism in the evolution of communication. In *Artificial life IV*, pages 40–48. Cambridge, MA, 1994.

Mark R Adler, Alvah B Davis, Robert Weihmayer, and Ralph W Worrest. Conflict-resolution strategies for nonhierarchical distributed agents. In *Distributed artificial intelligence*, pages 139–161. Elsevier, 1989.

Sanjeevan Ahilan and Peter Dayan. Feudal multi-agent hierarchies for cooperative reinforcement learning. *arXiv preprint arXiv:1901.08492*, 2019.

David Andre and Astro Teller. Evolving team darwin united. In *RoboCup-98: Robot Soccer World Cup II 2*, pages 346–351. Springer, 1999.

Peter Auer, Nicolo Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine learning*, 47(2):235–256, 2002.

Robert Axelrod. Effective choice in the prisoner's dilemma. *Journal of conflict resolution*, 24(1):3–25, 1980.

Robert Axelrod and William D Hamilton. The evolution of cooperation. *science*, 211(4489): 1390–1396, 1981.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

Thomas Back. *Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms*. Oxford university press, 1996.

Pierre-Luc Bacon, Jean Harb, and Doina Precup. The option-critic architecture. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

Dzmitry Bahdanau, Kyung Hyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015*, 2015.

Dzmitry Bahdanau, Felix Hill, Jan Leike, Edward Hughes, Pushmeet Kohli, and Edward Grefenstette. Learning to understand goal specifications by modelling reward. In *International Conference on Learning Representations*, 2019.

Bowen Baker, Ingmar Kanitscheider, Todor Markov, Yi Wu, Glenn Powell, Bob McGrew, and Igor Mordatch. Emergent tool use from multi-agent autocurricula. In *International Conference on Learning Representations*, 2020.

Tucker Balch. Reward and diversity in multirobot foraging. 1999.

Tucker Balch et al. Learning roles: Behavioral diversity in robot teams. In *AAAI Workshop on Multiagent Learning*, 1997.

Tucker R Balch. *Behavioral diversity in learning robot teams*. Georgia Institute of Technology, 1998.

Albert Bandura and Richard H Walters. *Social learning theory*, volume 1. Englewood cliffs Prentice Hall, 1977.

Bikramjit Banerjee, Rajatish Mukherjee, and Sandip Sen. Learning mutual trust. In *Working Notes of AGENTS-00 Workshop on Deception, Fraud and Trust in Agent Societies*, pages 9–14, 2000.

Trapit Bansal, Jakub Pachocki, Szymon Sidor, Ilya Sutskever, and Igor Mordatch. Emergent complexity via multi-agent competition. *arXiv preprint arXiv:1710.03748*, 2017.

Nolan Bard, Jakob N Foerster, Sarath Chandar, Neil Burch, Marc Lanctot, H Francis Song, Emilio Parisotto, Vincent Dumoulin, Subhodeep Moitra, Edward Hughes, et al. The hanabi challenge: A new frontier for ai research. *Artificial Intelligence*, 280:103216, 2020.

Paul Barde, Julien Roy, Wonseok Jeon, Joelle Pineau, Chris Pal, and Derek Nowrouzezahrai. Adversarial soft advantage fitting: Imitation learning without policy optimization. *Advances in Neural Information Processing Systems*, 33:12334–12344, 2020.

Paul Barde, Tristan Karch, Derek Nowrouzezahrai, Clément Moulin-Frier, Christopher Pal, and Pierre-Yves Oudeyer. Learning to guide and to be guided in the architect-builder problem. In *International Conference on Learning Representations*, 2022. URL `https://openreview.net/forum?id=swiyAeGzFhQ`.

Paul Barde, Jakob Foerster, Derek Nowrouzezahrai, and Amy Zhang. A model-based solution to the offline multi-agent reinforcement learning coordination problem, 2023.

Jeffrey A Barrett and Brian Skyrms. Self-assembling games. *The British Journal for the Philosophy of Science*, 2017.

Samuel Barrett, Peter Stone, and Sarit Kraus. Empirical evaluation of ad hoc teamwork in the pursuit domain. In *The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 2*, pages 567–574, 2011.

Sean L Barton, Nicholas R Waytowich, Erin Zaroukian, and Derrik E Asher. Measuring collaborative emergent behavior in multi-agent reinforcement learning. In *International Conference on Human Systems Engineering and Design: Future Trends and Applications*, pages 422–427. Springer, 2018.

Randall D Beer. A dynamical systems perspective on agent-environment interaction. *Artificial intelligence*, 72(1-2):173–215, 1995.

Richard Bellman. Dynamic programming. *Science*, 153(3731):34–37, 1966.

Miroslav Benda. On optimal cooperation of knowledge sources. *Techical Report CCS-C2010-28*, 1985.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48, 2009.

H Berenji and David Vengerov. Learning, cooperation, and coordination in multi-agent systems. *Intelligent Inference Systems Corporation, Technical report*, 2000a.

Hamid R Berenji and David Vengerov. Advantages of cooperation between reinforcement learning agents in difficult stochastic problems. In *Ninth IEEE International Conference on Fuzzy Systems. FUZZ-IEEE 2000 (Cat. No. 00CH37063)*, volume 2, pages 871–876. IEEE, 2000b.

Daniel S Bernstein, Robert Givan, Neil Immerman, and Shlomo Zilberstein. The complexity of decentralized control of markov decision processes. *Mathematics of operations research*, 27(4):819–840, 2002.

Katrien Beuls and Luc Steels. Agent-based models of strategies for the emergence and evolution of grammatical agreement. *PloS one*, 8(3):e58960, 2013.

H Joseph Blumenthal and Gary B Parker. Co-evolving team capture strategies for dissimilar robots. In *AAAI Technical Report (2)*, pages 15–22, 2004.

Josh C Bongard. The legion system: A novel approach to evolving heterogeneity for collective problem solving. In *European Conference on Genetic Programming*, pages 16–28. Springer, 2000.

Tilman Börgers and Daniel Krahmer. *An introduction to the theory of mechanism design*. Oxford University Press, USA, 2015.

Craig Boutilier. Learning conventions in multiagent stochastic domains using likelihood estimates. In *Proceedings of the Twelfth international conference on Uncertainty in artificial intelligence*, pages 106–114, 1996a.

Craig Boutilier. Planning, learning and coordination in multiagent decision processes. In *TARK*, volume 96, pages 195–210. Citeseer, 1996b.

Michael Bowling. Convergence problems of general-sum multiagent reinforcement learning. In *ICML*, pages 89–94, 2000.

Michael Bowling and Manuela Veloso. An analysis of stochastic game theory for multiagent reinforcement learning. Technical report, Carnegie-Mellon Univ Pittsburgh Pa School of Computer Science, 2000.

Michael Bowling and Manuela Veloso. Rational and convergent learning in stochastic games. In *International joint conference on artificial intelligence*, volume 17, pages 1021–1026. Citeseer, 2001.

Justin Boyan and Michael Littman. Packet routing in dynamically changing networks: A reinforcement learning approach. *Advances in neural information processing systems*, 6, 1993.

Brian Boyd. The evolution of stories: from mimesis to language, from fact to fiction. *Wiley Interdisciplinary Reviews: Cognitive Science*, 9(1):e1444, 2018.

Robert Boyd and Peter J Richerson. Culture and the evolution of human cooperation. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 364(1533):3281–3288, 2009.

Robert Boyd, Peter J Richerson, and Joseph Henrich. The cultural niche: Why social learning is essential for human adaptation. *Proceedings of the National Academy of Sciences*, 108(supplement_2):10918–10925, 2011.

Ronen Brafman and Moshe Tennenholtz. Efficient learning equilibrium. *Advances in Neural Information Processing Systems*, 15, 2002.

Wilfried Brauer and Gerhard Weiß. Multi-machine scheduling-a multi-agent learning approach. In *Proceedings International Conference on Multi Agent Systems (Cat. No. 98EX160)*, pages 42–48. IEEE, 1998.

Cameron B Browne, Edward Powley, Daniel Whitehouse, Simon M Lucas, Peter I Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton. A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in games*, 4(1):1–43, 2012.

Jerome Bruner. Child's talk: Learning to use language. *Child Language Teaching and Therapy*, 1(1):111–114, 1985.

Larry Bull. Evolutionary computation in multi-agent environments: Partners. In *Proc. 7th Int. Conf. Genetic Algorithms (ICGA 97)*, 1997.

Larry Bull. Evolutionary computing in multi-agent environments: Operators. In *Evolutionary Programming VII: 7th International Conference, EP98 San Diego, California, USA, March 25–27, 1998 Proceedings 7*, pages 43–52. Springer, 1998.

Lawrence Bull and Terence C Fogarty. Evolving cooperative communicating classifier systems. In *Proceedings of the Fourth Annual Conference on Evolutionary Programming (EP94)*, pages 308–315. World Scientific, 1994.

Stefan Bussmann and Jorg Muller. A negotiation framework for co-operating agents. *Proceedings of CKBS-SIG*, pages 1–17, 1992.

M Busuioc and C Winter. Negotiation and intelligent agents. *Project NOMADS-001, BT Laboratories internal report*, 1995.

Stephanie Cammarata, David McArthur, and Randall Steeb. Strategies of cooperation in distributed problem solving. In *Readings in Distributed Artificial Intelligence*, pages 102–105. Elsevier, 1988.

Kris Cao, Angeliki Lazaridou, Marc Lanctot, Joel Z Leibo, Karl Tuyls, and Stephen Clark. Emergent communication through negotiation. *arXiv preprint arXiv:1804.03980*, 2018.

David Carmel and Shaul Markovitch. The m* algorithm: Incorporating opponent models into adversary search. In *Technical Report CIS9402*. Technion Haifa, Israel, 1994.

Micah Carroll, Rohin Shah, Mark K Ho, Tom Griffiths, Sanjit Seshia, Pieter Abbeel, and Anca Dragan. On the utility of learning about humans for human-ai coordination. *Advances in neural information processing systems*, 32, 2019.

Thomas Cederborg and Pierre-Yves Oudeyer. A social learning formalism for learners trying to figure out what a teacher wants them to do. *Paladyn: Journal of Behavioral Robotics*, 5:64–99, 2014.

Lars-Erik Cederman. *Emergent actors in world politics: how states and nations develop and dissolve*, volume 2. Princeton University Press, 1997.

Rahma Chaabouni, Eugene Kharitonov, Diane Bouchacourt, Emmanuel Dupoux, and Marco Baroni. Compositionality and Generalization in Emergent Languages. In *ACL 2020 - 8th annual meeting of the Association for Computational Linguistics*, Seattle / Virtual, United States, July 2020. URL `https://hal.science/hal-02959466`.

Jhelum Chakravorty, Nadeem Ward, Julien Roy, Maxime Chevalier-Boisvert, Sumana Basu, Andrei Lupu, and Doina Precup. Option-critic in cooperative multi-agent systems. *arXiv preprint arXiv:1911.12825*, 2019.

Georgios Chalkiadakis and Craig Boutilier. Coordination in multiagent reinforcement learning: A bayesian approach. In *Proceedings of the second international joint conference on Autonomous agents and multiagent systems*, pages 709–716, 2003.

Yu-Han Chang, Tracey Ho, and Leslie Kaelbling. All learning is local: Multi-agent learning in global reward games. *Advances in neural information processing systems*, 16, 2003.

Sikai Chen, Jiqian Dong, Paul Ha, Yujie Li, and Samuel Labi. Graph neural network and reinforcement learning for multi-agent cooperative control of connected autonomous vehicles. *Computer-Aided Civil and Infrastructure Engineering*, 36(7):838–857, 2021.

Nuttapong Chentanez, Andrew G Barto, and Satinder P Singh. Intrinsically motivated reinforcement learning. In *Advances in neural information processing systems*, pages 1281–1288, 2005.

Kuan-Jung Chiang, Dimitra Emmanouilidou, Hannes Gamper, David Johnston, Mihai Jalobeanu, Edward Cutrell, Andrew Wilson, Winko W. An, and Ivan Tashev. A closed-loop adaptive brain-computer interface framework: Improving the classifier with the use of error-related potentials. In *2021 10th International IEEE/EMBS Conference on Neural Engineering (NER)*, pages 487–490, 2021. doi: 10.1109/NER49283.2021.9441133.

Valliappa Chockalingam, Tegg Tae Kyong Sung, Feryal Behbahani, Rishab Gargeya, Amlesh Sivanantham, and Aleksandra Malysheva. Extending world models for multi-agent reinforcement learning in malmö. In *AIIDE Workshops*, 2018.

Caroline Claus and Craig Boutilier. The dynamics of reinforcement learning in cooperative multiagent systems. *AAAI/IAAI*, 1998(746-752):2, 1998.

Scott H Clearwater, Bernardo A Huberman, and Tad Hogg. Cooperative solution of constraint satisfaction problems. *Science*, 254(5035):1181–1183, 1991.

Dave Cliff and Geoffrey F Miller. Tracking the red queen: Measurements of adaptive progress in co-evolutionary simulations. In *Advances in Artificial Life: Third European Conference on Artificial Life Granada, Spain, June 4–6, 1995 Proceedings 3*, pages 200–218. Springer Berlin Heidelberg, 1995.

Cédric Colas, Tristan Karch, Nicolas Lair, Jean-Michel Dussoux, Clément Moulin-Frier, Peter Dominey, and Pierre-Yves Oudeyer. Language as a cognitive tool to imagine goals in curiosity driven exploration. *Advances in Neural Information Processing Systems*, 33: 3761–3774, 2020.

Cédric Colas, Tristan Karch, Clément Moulin-Frier, and Pierre-Yves Oudeyer. Language as a Cognitive Tool: Dall-E, Humans and Vygotskian RL Agents, March 2021. URL `https://hal.archives-ouvertes.fr/hal-03159786`.

John Collins, Ned Hall, and Laurie Ann Paul. *Causation and counterfactuals*. Mit Press, 2004.

Susan E Conry, Robert A Meyer, and Victor R Lesser. Multistage negotiation in distributed planning. In *Readings in distributed artificial intelligence*, pages 367–384. Elsevier, 1988.

Daniel D Corkill. Hierarchical planning in a distributed environment. In *IJCAI*, volume 79, pages 168–175, 1979.

Kenneth James Williams Craik. *The nature of explanation*, volume 445. CUP Archive, 1967.

Vincent P Crawford and Joel Sobel. Strategic information transmission. *Econometrica: Journal of the Econometric Society*, pages 1431–1451, 1982.

Valentino Crespi, George Cybenko, Daniela Rus, and Massimo Santini. Decentralized control for coordinated flow of multi-agent systems. In *Proceedings of the 2002 International Joint Conference on Neural Networks. IJCNN'02 (Cat. No. 02CH37290)*, volume 3, pages 2604–2609. IEEE, 2002.

Randall Davis and Reid G Smith. Negotiation as a metaphor for distributed problem solving. *Artificial intelligence*, 20(1):63–109, 1983.

Richard Dawkins and John R Krebs. Animal signals: information or manipulation? 1978.

Peter Dayan and Geoffrey E Hinton. Feudal reinforcement learning. In *Advances in neural information processing systems*, pages 271–278, 1993.

Christian Schroeder de Witt, Tarun Gupta, Denys Makoviichuk, Viktor Makoviychuk, Philip HS Torr, Mingfei Sun, and Shimon Whiteson. Is independent learning all you need in the starcraft multi-agent challenge? *arXiv preprint arXiv:2011.09533*, 2020.

Megan T. deBettencourt, Jonathan D. Cohen, Ray F. Lee, Kenneth A. Norman, and Nicholas B. Turk-Browne. Closed-loop training of attention with real-time brain imaging. *Nature neuroscience*, 18:470 – 475, 2015.

Thomas Degris, Martha White, and Richard S Sutton. Off-policy actor-critic. In *Proceedings of the 29th International Conference on Machine Learning (ICML)*, pages 179–186, 2012.

Xiaotie Deng, Ningyuan Li, David Mguni, Jun Wang, and Yaodong Yang. On the complexity of computing markov perfect equilibrium in general-sum stochastic games. *National Science Review*, 10(1):nwac256, 2023.

Daniel C Dennett. Why the law of effect will not go away. *Journal for the Theory of Social Behaviour*, 1975.

Michael Dennis, Natasha Jaques, Eugene Vinitsky, Alexandre Bayen, Stuart Russell, Andrew Critch, and Sergey Levine. Emergent complexity and zero-shot transfer via unsupervised environment design. *Advances in neural information processing systems*, 33:13049–13061, 2020.

Jörg Denzinger and Matthias Fuchs. *Experiments in learning prototypical situations for variants of the pursuit game*. Technische Universität Kaiserslautern, Fachbereich Informatik, 1999.

Jean-Louis Dessalles, Jean Pierre Müller, and Denis Phan. Emergence in multi-agent systems: conceptual and methodological issues. 2007.

Yiming Ding, Carlos Florensa, Pieter Abbeel, and Mariano Phielipp. Goal-conditioned imitation learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 15298–15309, 2019.

Kurt Dresner and Peter Stone. Multiagent traffic management: A reservation-based intersection control mechanism. In *Autonomous Agents and Multiagent Systems, International Joint Conference on*, volume 3, pages 530–537. Citeseer, 2004.

Daniel C Dunn, Autumn-Lynn Harrison, Corrie Curtice, Sarah DeLand, Ben Donnelly, EI Fujioka, Eleanor Heywood, Connie Y Kot, Sarah Poulin, Meredith Whitten, et al. The importance of migratory connectivity for global ocean policy. *Proceedings of the Royal Society B*, 286(1911):20191472, 2019.

Edmund H Durfee and Victor R Lesser. Using partial global plans to coordinate distributed problem solvers. In *Readings in distributed artificial intelligence*, pages 285–293. Elsevier, 1988.

Edmund H Durfee, Victor R Lesser, and Daniel D Corkill. Coherent cooperation among communicating problem solvers. *IEEE Transactions on computers*, 100(11):1275–1291, 1987.

Edmund H Durfee, Victor R Lesser, and Daniel D Corkill. Trends in cooperative distributed problem solving. *IEEE Transactions on knowledge and data Engineering*, 1989.

Vladimir Egorov and Alexei Shpilman. Scalable multi-agent model-based reinforcement learning. In *Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems*, pages 381–390, 2022.

Jeffrey L Elman. Learning and development in neural networks: The importance of starting small. *Cognition*, 48(1):71–99, 1993.

Joseph Farrell. Cheap talk, coordination, and entry. *The RAND Journal of Economics*, pages 34–39, 1987.

Jacques Ferber and Gerhard Weiss. *Multi-agent systems: an introduction to distributed artificial intelligence*, volume 1. Addison-wesley Reading, 1999.

Sevan G Ficici and Jordan B Pollack. Challenges in coevolutionary learning: Arms-race dynamics, open-endedness, and mediocre stable states. In *Proceedings of the sixth international conference on Artificial life*, pages 238–247. MIT Press Cambridge, MA, 1998.

Sevan G Ficici and Jordan B Pollack. A game-theoretic approach to the simple coevolutionary algorithm. In *Parallel Problem Solving from Nature PPSN VI: 6th International Conference Paris, France, September 18–20, 2000 Proceedings 6*, pages 467–476. Springer, 2000.

Chelsea Finn, Paul Christiano, Pieter Abbeel, and Sergey Levine. A connection between generative adversarial networks, inverse reinforcement learning, and energy-based models. *arXiv preprint arXiv:1611.03852*, 2016a.

Chelsea Finn, Sergey Levine, and Pieter Abbeel. Guided cost learning: Deep inverse optimal control via policy optimization. In *Proceedings of the 33rd International Conference on Machine Learning (ICML)*, pages 49–58, 2016b.

K Fischer, N Kuhn, HJ Muller, JP Muller, and M Pischel. Sophisticated and distributed: The transportation domain. In *Proceedings of 9th IEEE Conference on Artificial Intelligence for Applications*, page 454. IEEE, 1993.

Jakob Foerster, Ioannis Alexandros Assael, Nando de Freitas, and Shimon Whiteson. Learning to communicate with deep multi-agent reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 2137–2145, 2016.

Jakob Foerster, Francis Song, Edward Hughes, Neil Burch, Iain Dunning, Shimon Whiteson, Matthew Botvinick, and Michael Bowling. Bayesian action decoder for deep multi-agent reinforcement learning. *International Conference on Machine Learning*, 2019.

Jakob N Foerster, Richard Y Chen, Maruan Al-Shedivat, Shimon Whiteson, Pieter Abbeel, and Igor Mordatch. Learning with opponent-learning awareness. *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 17, 2018a.

Jakob N Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. Counterfactual multi-agent policy gradients. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018b.

Sébastien Forestier, Rémy Portelas, Yoan Mollard, and Pierre-Yves Oudeyer. Intrinsically motivated goal exploration processes with automatic curriculum learning. *Journal of Machine Learning Research*, 2022.

Justin Fu, Katie Luo, and Sergey Levine. Learning robust rewards with adversarial inverse reinforcement learning. In *Proceedings of the 5th International Conference on Learning Representations (ICLR)*, 2017.

Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4rl: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*, 2020.

Agustin Fuentes. Human niche, human behaviour, human nature. *Interface Focus*, 7(5): 20160136, 2017.

Scott Fujimoto and Shixiang Shane Gu. A minimalist approach to offline reinforcement learning. *Advances in neural information processing systems*, 34:20132–20145, 2021.

Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. In *International conference on machine learning*, pages 2052–2062. PMLR, 2019.

Bruno Galantucci and Simon Garrod. Experimental semiotics: a review. *Frontiers in human neuroscience*, 5:11, 2011.

Bruno Galantucci and Luc Steels. The emergence of embodied communication in artificial agents and humans. *Embodied communication in humans and machines*, pages 229–256, 2008.

Lei Gao, Yao-Tang Li, and Rui-Wu Wang. The shift between the red queen and the red king effects in mutualisms. *Scientific reports*, 5(1):8237, 2015.

Andrew Garland and Richard Alterman. Autonomous agents that learn to better coordinate. *Autonomous Agents and Multi-Agent Systems*, 8(3):267–301, 2004.

Michael Georgeff. Communication and interaction in multi-agent planning. In *Readings in distributed artificial intelligence*, pages 200–204. Elsevier, 1988a.

Michael Georgeff. A theory of action for multiagent planning. In *Readings in distributed artificial intelligence*, pages 205–209. Elsevier, 1988b.

Seyed Kamyar Seyed Ghasemipour, Richard Zemel, and Shixiang Gu. A divergence minimization perspective on imitation learning methods. In *Proceedings of the 3rd Conference on Robot Learning (CoRL)*, 2019.

Mohammad Ghavamzadeh and Sridhar Mahadevan. Learning to communicate and act using hierarchical reinforcement learning. *Computer Science Department Faculty Publication Series*, page 172, 2004.

Natalie S Glance and Bernardo A Huberman. The dynamics of social dilemmas. *Scientific American*, 270(3):76–81, 1994.

Piotr Jan Gmytrasiewicz. *A decision-theoretic model of coordination and communication in autonomous systems*. University of Michigan, 1992.

BM Good. Evolving multi-agent systems: Comparing existing approaches and suggesting new directions. *Masters's thesis, University of Sussex*, 2000.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 2672–2680, 2014.

Michael A Goodrich and Alan C Schultz. *Human-robot interaction: a survey*. Now Publishers Inc, 2008.

Maria Gordin, Sandip Sen, and Narendra Puppala. Evolving cooperative groups: Preliminary results. In *Proc. of the AAAI-97 Workshop on Multi-Agent Learning*, 1997.

Stephen Grand and Dave Cliff. Creatures: Entertainment software agents with artificial life. *Autonomous Agents and Multi-Agent Systems*, 1:39–57, 1998.

Stephen Grand, Dave Cliff, and Anil Malhotra. Creatures: Artificial life autonomous software agents for home entertainment. In *Proceedings of the first international conference on Autonomous agents*, pages 22–29, 1997.

Amy Greenwald, Keith Hall, Roberto Serrano, et al. Correlated q-learning. In *ICML*, volume 3, pages 242–249, 2003.

Jonathan Grizou, Manuel Lopes, and Pierre-Yves Oudeyer. Robot learning simultaneously a task and how to interpret human instructions. In *2013 IEEE third joint international conference on development and learning and epigenetic robotics (ICDL)*, pages 1–8. IEEE, 2013.

Jonathan Grizou, Inaki Iturrate, Luis Montesano, Pierre-Yves Oudeyer, and Manuel Lopes. Calibration-free bci based control. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 28, 2014a.

Jonathan Grizou, Inaki Iturrate, Luis Montesano, Pierre-Yves Oudeyer, and Manuel Lopes. Interactive learning from unlabeled instructions. In *Proceedings of the Thirtieth Conference on Uncertainty in Artificial Intelligence (UAI)*, number CONF, 2014b.

Carlos Guestrin, Michail Lagoudakis, and Ronald Parr. Coordinated reinforcement learning. In *ICML*, volume 2, pages 227–234. Citeseer, 2002.

Philip Hugh Gulliver. *Disputes and negotiations: A cross-cultural perspective*. Academic Press, 1979.

Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of Wasserstein GANs. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 5767–5777, 2017.

Jayesh K. Gupta, Maxim Egorov, and Mykel J. Kochenderfer. Cooperative multi-agent control using deep reinforcement learning. In *AAMAS Workshops*, 2017.

S Gupta and A Dukkipati. Winning an election: On emergent strategic communication in multi-agent networks. In *Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems, AAMAS*, volume 2020, pages 1861–1863. International Foundation for Autonomous Agents and Multiagent Systems (IFAAMAS), 2020.

Izzeddin Gur, Natasha Jaques, Yingjie Miao, Jongwook Choi, Manoj Tiwari, Honglak Lee, and Aleksandra Faust. Environment generation for zero-shot compositional reinforcement learning. *Advances in Neural Information Processing Systems*, 34:4157–4169, 2021.

David Ha and Jürgen Schmidhuber. World models. *arXiv preprint arXiv:1803.10122*, 2018.

Tuomas Haarnoja, Haoran Tang, Pieter Abbeel, and Sergey Levine. Reinforcement learning with deep energy-based policies. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, pages 1352–1361, 2017.

Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, et al. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 2018.

Dylan Hadfield-Menell, Anca Dragan, Pieter Abbeel, and Stuart Russell. Cooperative inverse reinforcement learning. *arXiv preprint arXiv:1606.03137*, 2016.

William D Hamilton. The genetical evolution of social behaviour. ii. *Journal of theoretical biology*, 7(1):17–52, 1964.

Akira Hara. Emergence of cooperative behavior using adg; automatically defined groups. In *GECCO-99: Proc. Genetic and Evolutionary Computation Conference*, pages 1039–1046. Morgan Kaufmann, 1999.

Inman Harvey, Phil Husbands, Dave Cliff, Adrian Thompson, and Nick Jakobi. Evolutionary robotics: the sussex approach. *Robotics and autonomous systems*, 20(2-4):205–224, 1997.

Steven C Hayes and Brandon T Sanford. Cooperation came first: Evolution and human cognition. *Journal of the Experimental Analysis of Behavior*, 101(1):112–129, 2014.

Barbara Hayes-Roth. A blackboard architecture for control. *Artificial intelligence*, 26(3): 251–321, 1985.

Thomas Haynes and Sandip Sen. Evolving behavioral strategies in predators and prey. In *Adaption and Learning in Multi-Agent Systems: IJCAI'95 Workshop Montréal, Canada, August 21, 1995 Proceedings 14*, pages 113–126. Springer, 1996.

Thomas Haynes, Sandip Sen, Dale Schoenefeld, and Roger Wainwright. Evolving a team. In *Working notes for the AAAI symposium on genetic programming*, pages 23–30. MIT Cambridge, 1995a.

Thomas Haynes, Sandip Sen, Dale Schoenefeld, and Roger Wainwright. Evolving multiagent coordination strategies with genetic programming. *Artificial Intelligence*, 1995b.

Thomas Haynes, Kit Lau, and Sandip Sen. Learning cases to compliment rules for conflict resolution in multiagent systems. In *Adaptation, Coevolution and Learning in Multiagent Systems: Papers from the 1996 AAAI Spring Symposium*, pages 51–56, 1996.

Thomas Haynes, Sandip Sen, et al. Crossover operators for evolving a team. *Genetic programming*, 199, 1997.

He He, Jordan Boyd-Graber, Kevin Kwok, and Hal Daumé III. Opponent modeling in deep reinforcement learning. In *International Conference on Machine Learning*, pages 1804–1813, 2016.

Pablo Hernandez-Leal, Bilal Kartal, and Matthew E. Taylor. Agent Modeling as Auxiliary Task for Deep Reinforcement Learning. In *AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, 2019.

Edward Allen Herre, Nancy Knowlton, Ulrich Gerhard Mueller, and Stuart A Rehner. The evolution of mutualisms: exploring the paths between conflict and cooperation. *Trends in ecology & evolution*, 14(2):49–53, 1999.

Todd Hester, Michael Quinlan, and Peter Stone. Generalized model learning for reinforcement learning on a humanoid robot. In *2010 IEEE International Conference on Robotics and Automation*, pages 2369–2374. IEEE, 2010.

W Daniel Hillis. Co-evolving parasites improve simulated evolution as an optimization procedure. *Physica D: Nonlinear Phenomena*, 42(1-3):228–234, 1990.

Robert A Hinde. Animal signals: Ethological and games-theory approaches are not incompatible. *Animal Behaviour*, 29(2):535–542, 1981.

Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 4565–4573, 2016.

Zhang-Wei Hong, Shih-Yang Su, Tzu-Yun Shann, Yi-Hsiang Chang, and Chun-Yi Lee. A deep policy inference q-network for multi-agent systems. *arXiv preprint arXiv:1712.07893*, 2017.

Ronald A Howard. Dynamic programming and markov processes. 1960.

Hengyuan Hu, Adam Lerer, Alex Peysakhovich, and Jakob Foerster. "other-play" for zero-shot coordination. In *International Conference on Machine Learning*, pages 4399–4410. PMLR, 2020.

Junling Hu and Michael P Wellman. Self-fulfilling bias in multiagent learning. In *Proceedings of the Second International Conference on Multi-Agent Systems (ICMAS-96)*, pages 118–125, 1996.

Junling Hu and Michael P Wellman. Online learning about other agents in a dynamic multiagent system. In *Proceedings of the second international conference on Autonomous agents*, pages 239–246, 1998.

Junling Hu and Michael P Wellman. Nash q-learning for general-sum stochastic games. *Journal of machine learning research*, 4(Nov):1039–1069, 2003.

Junling Hu, Michael P Wellman, et al. Multiagent reinforcement learning: theoretical framework and an algorithm. In *ICML*, volume 98, pages 242–250, 1998.

Jun Huang, Nicholas R Jennings, and John Fox. An agent architecture for distributed medical care. In *Intelligent Agents: ECAI-94 Workshop on Agent Theories, Architectures, and Languages Amsterdam, The Netherlands August 8–9, 1994 Proceedings 1*, pages 219–232. Springer, 1995.

M Huhns and Munindar P Singh. Ckbs-94 tutorial: Distributed artificial intelligence for information systems. *Dake Centre, University of Keele*, 1994.

Hitoshi Iba. Emergent cooperation for multiple agents using genetic programming. In *International conference on parallel problem solving from nature*, pages 32–41. Springer, 1996.

Hitoshi Iba. Evolutionary learning of communicating agents. *Information Sciences*, 108(1-4): 181–205, 1998.

Shariq Iqbal and Fei Sha. Actor-attention-critic for multi-agent reinforcement learning. In *International Conference on Machine Learning*, pages 2961–2970, 2019.

Akira Ito. How do selfish agents learn to cooperate. In *Artificial Life V: Proceedings of the Fifth International Workshop on the Synthesis and Simulation of Living Systems*, volume 5, page 185. MIT Press, 1997.

Max Jaderberg, Wojciech M Czarnecki, Iain Dunning, Luke Marris, Guy Lever, Antonio Garcia Castaneda, Charles Beattie, Neil C Rabinowitz, Ari S Morcos, Avraham Ruderman, et al. Human-level performance in 3d multiplayer games with population-based reinforcement learning. *Science*, 364(6443):859–865, 2019.

Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparametrization with gumble-softmax. In *International Conference on Learning Representations (ICLR 2017)*. OpenReview. net, 2017.

Michael Janner, Justin Fu, Marvin Zhang, and Sergey Levine. When to trust your model: Model-based policy optimization. *Advances in neural information processing systems*, 32, 2019.

Thomas Jansen and R Paul Wiegand. Exploring the explorative advantage of the cooperative coevolutionary (1+ 1) ea. In *Genetic and Evolutionary Computation Conference*, pages 310–321. Springer, 2003.

Pieter Jan't Hoen and Karl Tuyls. Analyzing multi-agent reinforcement learning using evolutionary dynamics. In *Machine Learning: ECML 2004: 15th European Conference on Machine Learning, Pisa, Italy, September 20-24, 2004. Proceedings 15*, pages 168–179. Springer, 2004.

Natasha Jaques. *Social and Affective Machine Learning*. PhD thesis, Massachusetts Institute of Technology, 2019.

Natasha Jaques, Angeliki Lazaridou, Edward Hughes, Caglar Gulcehre, Pedro Ortega, Dj Strouse, Joel Z Leibo, and Nando De Freitas. Social influence as intrinsic motivation for multi-agent deep reinforcement learning. In *International Conference on Machine Learning*, pages 3040–3049, 2019.

Nick R Jennings. Coordination techniques for distributed artificial intelligence. 1996.

Wonseok Jeon, Paul Barde, Derek Nowrouzezahrai, and Joelle Pineau. Scalable multi-agent inverse reinforcement learning via actor-attention-critic. *arXiv preprint arXiv:2002.10525*, 2020.

Jiechuan Jiang and Zongqing Lu. Learning attentional communication for multi-agent cooperation. In *Advances in Neural Information Processing Systems*, pages 7254–7264, 2018.

Jiechuan Jiang and Zongqing Lu. Offline decentralized multi-agent reinforcement learning. *arXiv preprint arXiv:2108.01832*, 2021.

Kam-Chuen Jim and C Lee Giles. Talking helps: Evolving communicating agents for the predator-prey pursuit problem. *artificial life*, 6(3):237–254, 2000.

Yan Jin and Takeo Koyama. Multiagent planning through expectation based negotiation. In *Proceedings of the 10th Int Workshop on DAI, Texas*, 1990.

C Johnson, RR Reisinger, A Friedlaender, D Palacios, A Willson, A Zerbini, and M Lancaster. Protecting blue corridors–challenges and solutions for migratory whales navigating national and international seas. *WWF International*, 2022.

Leslie Pack Kaelbling, Michael L Littman, and Andrew W Moore. Reinforcement learning: A survey. *Journal of artificial intelligence research*, 4:237–285, 1996.

Spiros Kapetanakis and Daniel Kudenko. Improving on the reinforcement learning of coordination in cooperative multi-agent systems. In *Second AISB Symposium on Adaptive Agents and Multi-Agent Systems*. Citeseer, 2002a.

Spiros Kapetanakis and Daniel Kudenko. Reinforcement learning of coordination in cooperative multi-agent systems. *AAAI/IAAI*, 2002:326–331, 2002b.

Kapil D. Katyal, Matthew S. Johannes, Spencer Kellis, Tyson Aflalo, Christian Klaes, Timothy G. McGee, Matthew P. Para, Ying Shi, Brian Lee, Kelsie Pejsa, Charles Liu, Brock A. Wester, Francesco Tenore, James D. Beaty, Alan D. Ravitz, Richard A. Andersen, and Michael P. McLoughlin. A collaborative bci approach to autonomous control of a prosthetic

limb system. In *2014 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 1479–1482, 2014. doi: 10.1109/SMC.2014.6974124.

Paul E Kearney, Arvindra Sehmi, and Robert M Smith. Emergent behaviour in a multi-agent economic situation. In *Proceedings of the 11th European Conference on Artificial Intelligence*, pages 284–288, 1994.

Graham Kendall and Mark Willdig. An investigation of an adaptive poker player. In *AI 2001: Advances in Artificial Intelligence: 14th Australian Joint Conference on Artificial Intelligence Adelaide, Australia, December 10–14, 2001 Proceedings 14*, pages 189–200. Springer, 2001.

Rahul Kidambi, Aravind Rajeswaran, Praneeth Netrapalli, and Thorsten Joachims. Morel: Model-based offline reinforcement learning. *Advances in neural information processing systems*, 33:21810–21823, 2020.

Celeste Kidd, Steven T Piantadosi, and Richard N Aslin. The goldilocks effect: Human infants allocate attention to visual sequences that are neither too simple nor too complex. *PloS one*, 7(5):e36399, 2012.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Julia Kiseleva, Ziming Li, Mohammad Aliannejadi, Shrestha Mohanty, Maartje ter Hoeve, Mikhail Burtsev, Alexey Skrynnik, Artem Zholus, Aleksandr Panov, Kavya Srinet, et al. Neurips 2021 competition iglu: Interactive grounded language understanding in a collaborative environment. *arXiv preprint arXiv:2110.06536*, 2021.

Hiroaki Kitano, Minoru Asada, Yasuo Kuniyoshi, Itsuki Noda, and Eiichi Osawa. Robocup: The robot world cup initiative. In *Proceedings of the first international conference on Autonomous agents*, pages 340–347, 1997.

Levente Kocsis and Csaba Szepesvári. Bandit based monte-carlo planning. In *European conference on machine learning*, pages 282–293. Springer, 2006.

Ilya Kostrikov, Kumar Krishna Agrawal, Debidatta Dwibedi, Sergey Levine, and Jonathan Tompson. Discriminator-actor-critic: Addressing sample inefficiency and reward bias in adversarial imitation learning. In *Proceedings of the 7th International Conference on Learning Representations (ICLR)*, 2019.

Ilya Kostrikov, Ofir Nachum, and Jonathan Tompson. Imitation learning via off-policy distribution matching. In *Proceedings of the 8th International Conference on Learning Representations (ICLR)*, 2020.

Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit q-learning. *arxiv*, 2021.

Landon Kraemer and Bikramjit Banerjee. Multi-agent reinforcement learning as a rehearsal for decentralized planning. *Neurocomputing*, 190:82–94, 2016.

Thomas Kreifelts. A negotiation framework for autonomous agents. *Decentralized AI 2*, pages 71–88, 1991.

Alex Kuefler, Jeremy Morton, Tim Wheeler, and Mykel Kochenderfer. Imitating driver behavior with generative adversarial networks. In *Proceedings of 2017 IEEE Intelligent Vehicles Symposium (IV)*, pages 204–211, 2017.

Tejas D Kulkarni, Karthik Narasimhan, Ardavan Saeedi, and Josh Tenenbaum. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. *Advances in neural information processing systems*, 29:3675–3683, 2016.

Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 33: 1179–1191, 2020.

Karol Kurach, Anton Raichuk, Piotr Stańczyk, Michał Zajac, Olivier Bachem, Lasse Espeholt, Carlos Riquelme, Damien Vincent, Marcin Michalski, Olivier Bousquet, et al. Google research football: A novel reinforcement learning environment. *arXiv preprint arXiv:1907.11180*, 2019.

Martin Lauer. An algorithm for distributed reinforcement learning in cooperative multiagent systems. In *Proc. 17th International Conf. on Machine Learning*, 2000.

Angeliki Lazaridou and Marco Baroni. Emergent multi-agent communication in the deep learning era. *arXiv preprint arXiv:2006.02419*, 2020.

Angeliki Lazaridou, Alexander Peysakhovich, and Marco Baroni. Multi-agent cooperation and the emergence of (natural) language. *arXiv preprint arXiv:1612.07182*, 2016.

Angeliki Lazaridou, Alexander Peysakhovich, and Marco Baroni. Multi-agent cooperation and the emergence of (natural) language. In *International Conference on Learning Representations*, 2017. URL `https://openreview.net/forum?id=Hk8N3Sclg`.

Erwan Lecarpentier. *Reinforcement Learning in Non-Stationary Environments*. PhD thesis, Université de Toulouse & Institut Supérieur de l'Aéronautique et de l'Espace, 2020.

Adam Lerer and Alexander Peysakhovich. Maintaining cooperation in complex social dilemmas using deep reinforcement learning. *arXiv preprint arXiv:1707.01068*, 2017.

Adam Lerer and Alexander Peysakhovich. Learning existing social conventions via observationally augmented self-play. In *Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society*, pages 107–114, 2019.

Victor R Lesser and Daniel D Corkill. Functionally accurate, cooperative distributed systems. In *Readings in Distributed Artificial Intelligence*, pages 295–310. Elsevier, 1988.

Victor R Lesser and Daniel G Corkill. The distributed vehicle monitoring testbed: A tool for investigating distributed problem solving networks. *AI magazine*, 4(3):15–15, 1983.

Victor R Lesser, Daniel D Corkill, and Edmund H Durfee. An update on the distributed vehicle monitoring testbed, 1987.

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

Jiaoyang Li, Zhe Chen, Yi Zheng, Shao-Hung Chan, Daniel Harabor, Peter J Stuckey, Hang Ma, and Sven Koenig. Scalable rail planning and replanning: Winning the 2020 flatland challenge. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 31, pages 477–485, 2021.

Eric Liang and Richard Liaw. Scaling multi-agent reinforcement learning, 2018. URL `https://bair.berkeley.edu/blog/2018/12/12/rllib/`. Accessed on February 28, 2023.

Mark Irving Lichbach. *The cooperator's dilemma*. University of Michigan Press, 1996.

Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

Michael L Littman. Markov games as a framework for multi-agent reinforcement learning. In *Machine learning proceedings 1994*, pages 157–163. Elsevier, 1994.

Michael L Littman et al. Friend-or-foe q-learning in general-sum games. In *ICML*, volume 1, pages 322–328, 2001.

Haotian Liu and Wenchuan Wu. Online multi-agent reinforcement learning for decentralized inverter-based volt-var control. *IEEE Transactions on Smart Grid*, 12(4):2980–2990, 2021.

Iou-Jen Liu, Unnat Jain, Raymond A Yeh, and Alexander Schwing. Cooperative exploration for multi-agent deep reinforcement learning. In *International Conference on Machine Learning*, pages 6826–6836. PMLR, 2021.

Manuel Lopes, Thomas Cederbourg, and Pierre-Yves Oudeyer. Simultaneous acquisition of task and feedback models. In *2011 IEEE international conference on development and learning (ICDL)*, volume 2, pages 1–7. IEEE, 2011.

Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. In *Advances in Neural Information Processing Systems*, pages 6379–6390, 2017.

R Duncan Luce and Howard Raiffa. *Games and decisions: Introduction and critical survey*. Courier Corporation, 1989.

Sean Luke and Lee Spector. Evolving teamwork and coordination with genetic programming. *Genetic Programming*, 96:150–56, 1996.

Sean Luke, Charles Hohn, Jonathan Farris, Gary Jackson, and James Hendler. Co-evolving soccer softbot team coordination with genetic programming. In *RoboCup-97: Robot Soccer World Cup I 1*, pages 398–411. Springer, 1998a.

Sean Luke et al. Genetic programming produced competitive soccer softbot teams for robocup97. *Genetic Programming*, 1998:214–222, 1998b.

Xueguang Lyu, Yuchen Xiao, Brett Daley, and Christopher Amato. Contrasting centralized and decentralized critics in multi-agent reinforcement learning. *arXiv preprint arXiv:2102.04402*, 2021.

Sridhar Mahadevan. Average reward reinforcement learning: Foundations, algorithms, and empirical results. *Recent advances in reinforcement Learning*, pages 159–195, 1996.

Anuj Mahajan, Tabish Rashid, Mikayel Samvelyan, and Shimon Whiteson. Maven: Multi-agent variational exploration. In *Advances in Neural Information Processing Systems*, pages 7613–7624, 2019.

Rajbala Makar, Sridhar Mahadevan, and Mohammad Ghavamzadeh. Hierarchical multi-agent reinforcement learning. In *Proceedings of the fifth international conference on Autonomous agents*, pages 246–253, 2001.

Maja J Mataric. Learning to behave socially. *From animals to animats*, 3:453–462, 1994a.

Maja J Mataric. Reward functions for accelerated learning. In *Machine learning proceedings 1994*, pages 181–189. Elsevier, 1994b.

Maja J Mataric. Using communication to reduce locality in distributed multiagent learning. *Journal of experimental & theoretical artificial intelligence*, 10(3):357–369, 1998.

Maja J Mataric, Martin Nilsson, and Kristian T Simsarin. Cooperative multi-robot box-pushing. In *Proceedings 1995 IEEE/RSJ International Conference on Intelligent Robots and Systems. Human Robot Interaction and Cooperative Robots*, volume 3, pages 556–561. IEEE, 1995.

H Brendan McMahan and Geoffrey J Gordon. A unification of extensive-form games and markov decision processes. In *Proceedings of the National Conference on Artificial Intelligence*, volume 22, page 86. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2007.

Thomas Miconi. When evolving populations is better than coevolving individuals: The blind mice problem. In *IJCAI*, pages 647–652. Citeseer, 2003.

Jyoti Mishra and Adam Gazzaley. Closed-loop cognition: the next frontier arrives. *Trends in Cognitive Sciences*, 19:242–243, 2015.

Abdulla M Mohamed and Michael N Huhns. Multiagent benevolence as a societal norm. In *Social order in multiagent systems*, pages 65–83. Springer, 2001.

A. W. Moore and C. G. Atkeson. Prioritized sweeping: Reinforcement learning with less data and less time. *Machine Learning*, 13:103–130, 1993.

Igor Mordatch and Pieter Abbeel. Emergence of grounded compositional language in multi-agent populations. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

Jörg P Müller and Markus Pischel. An architecture for dynamically interacting agents. *International Journal of Intelligent and Cooperative Information Systems*, 3(01):25–45, 1994.

Manisha Mundhe and Sandip Sen. Evaluating concurrent reinforcement learners. In *Proceedings Fourth International Conference on MultiAgent Systems*, pages 421–422. IEEE, 2000a.

Manisha Mundhe and Sandip Sen. Evolving agent socienties that avoid social dilemmas. In *GECCO*, pages 809–816, 2000b.

Santiago Muñoz-Moldes and Axel Cleeremans. Delineating implicit and explicit processes in neurofeedback learning. *Neuroscience & Biobehavioral Reviews*, 118:681–688, 2020. ISSN 0149-7634. doi: https://doi.org/10.1016/j.neubiorev.2020.09.003. URL `https://www.sciencedirect.com/science/article/pii/S0149763420305595`.

Ofir Nachum, Shixiang Shane Gu, Honglak Lee, and Sergey Levine. Data-efficient hierarchical reinforcement learning. *Advances in Neural Information Processing Systems*, 31: 3303–3313, 2018a.

Ofir Nachum, Mohammad Norouzi, Kelvin Xu, and Dale Schuurmans. Trust-PCL: An off-policy trust region method for continuous control. In *Proceedings of the 6th International Conference on Learning Representations (ICLR)*, 2018b.

Ofir Nachum, Yinlam Chow, Bo Dai, and Lihong Li. DualDICE: Behavior-agnostic estimation of discounted stationary distribution corrections. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 2318–2328, 2019.

Yasuo Nagayuki, Shin Ishii, and Kenji Doya. Multi-agent reinforcement learning: An approach based on the other agent's internal model. In *Proceedings Fourth International Conference on MultiAgent Systems*, pages 215–221. IEEE, 2000.

Ranjit Nair, Milind Tambe, Makoto Yokoo, David Pynadath, and Stacy Marsella. Taming decentralized pomdps: Towards efficient policy computation for multiagent settings. In *IJCAI*, volume 3, pages 705–711, 2003.

Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814, 2010.

Anjali Narayan-Chen, Prashant Jayannavar, and Julia Hockenmaier. Collaborative dialogue in minecraft. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5405–5415, 2019.

Sanmit Narvekar, Bei Peng, Matteo Leonetti, Jivko Sinapov, Matthew E Taylor, and Peter Stone. Curriculum learning for reinforcement learning domains: A framework and survey. *The Journal of Machine Learning Research*, 21(1):7382–7431, 2020.

Kamal K Ndousse, Douglas Eck, Sergey Levine, and Natasha Jaques. Emergent social learning via multi-agent reinforcement learning. In *International Conference on Machine Learning*, pages 7991–8004. PMLR, 2021.

Andrew Y Ng, Daishi Harada, and Stuart Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *Icml*, volume 99, pages 278–287, 1999.

Andrew Y Ng, Stuart J Russell, et al. Algorithms for inverse reinforcement learning. In *Icml*, volume 1, page 2, 2000.

Khanh Nguyen, Dipendra Misra, Robert Schapire, Miro Dudík, and Patrick Shafto. Interactive learning from activity description, 2021.

Noam Nisan, Tim Roughgarden, Eva Tardos, and Vijay V Vazirani. *Algorithmic game theory*. Cambridge university press, 2007.

Michael Noukhovitch, Travis LaCroix, Angeliki Lazaridou, and Aaron Courville. Emergent communication under competition. In *Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems*, pages 974–982, 2021.

Martin A Nowak and David C Krakauer. The evolution of language. *Proceedings of the National Academy of Sciences*, 96(14):8028–8033, 1999.

Martin A Nowak and Karl Sigmund. Evolution of indirect reciprocity by image scoring. *Nature*, 393(6685):573–577, 1998.

Ann Nowe, Albert Verbeeck, and Tom Lenaerts. Learning agents in a homo egualis society. In *In: Proceedings of the Agents 01 Workshop on Learning Agents, page 76, Montreal, Canada. Agents 2001.*, 2001.

Luis Nunes and Eugénio Oliveira. Learning from multiple sources. In *AAMAS*, volume 4, pages 1106–1113, 2004.

Hyacinth S Nwana. Software agents: An overview. *The knowledge engineering review*, 11 (3):205–244, 1996.

Hyacinth S Nwana, L Lee, and Nicholas R Jennings. Co-ordination in multi-agent systems. *Software Agents and Soft Computing Towards Enhancing Machine Intelligence*, pages 42–58, 1997.

Junhyuk Oh, Yijie Guo, Satinder Singh, and Honglak Lee. Self-imitation learning. In *ICML*, 2018.

Frans A Oliehoek and Christopher Amato. A concise introduction to decentralized pomdps, 2015.

Frans A Oliehoek, Matthijs TJ Spaan, and Nikos Vlassis. Optimal and approximate q-value functions for decentralized pomdps. *Journal of Artificial Intelligence Research*, 32: 289–353, 2008.

OpenAI, Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemyslaw Debiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Chris Hesse, Rafal Józefowicz, Scott Gray, Catherine Olsson, Jakub Pachocki, Michael Petrov, Henrique Pondé de Oliveira Pinto, Jonathan Raiman, Tim Salimans, Jeremy Schlatter, Jonas Schneider, Szymon Sidor, Ilya Sutskever, Jie Tang, Filip Wolski, and Susan Zhang. Dota 2 with large scale deep reinforcement learning. 2019. URL `https://arxiv.org/abs/1912.06680`.

Esben H Ostergaard, Gaurav S Sukhatme, and Maja J Matari. Emergent bucket brigading: a simple mechanisms for improving performance in multi-robot constrained-space foraging tasks. In *Proceedings of the fifth international conference on Autonomous agents*, pages 29–30, 2001.

Praveen Palanisamy. Multi-agent connected autonomous driving using deep reinforcement learning. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–7. IEEE, 2020.

Ling Pan, Longbo Huang, Tengyu Ma, and Huazhe Xu. Plan better amid conservatism: Offline multi-agent reinforcement learning with actor rectification. In *International Conference on Machine Learning*, pages 17221–17237. PMLR, 2022.

Liviu Panait and Sean Luke. Cooperative multi-agent learning: The state of the art. *Autonomous agents and multi-agent systems*, 11(3):387–434, 2005.

Liviu Panait, R Paul Wiegand, and Sean Luke. Improving coevolutionary search for optimal multiagent behaviors. In *IJCAI*, pages 653–660, 2003.

Liviu Panait, R Paul Wiegand, and Sean Luke. A sensitivity analysis of a cooperative coevolutionary algorithm biased for optimization. In *Genetic and Evolutionary Computation–GECCO 2004: Genetic and Evolutionary Computation Conference, Seattle, WA, USA, June 26-30, 2004. Proceedings, Part I*, pages 573–584. Springer, 2004a.

Liviu Panait, R Paul Wiegand, and Sean Luke. A visual demonstration of convergence properties of cooperative coevolution. In *Parallel Problem Solving from Nature-PPSN VIII: 8th International Conference, Birmingham, UK, September 18-22, 2004. Proceedings 8*, pages 892–901. Springer, 2004b.

Christos H Papadimitriou and John N Tsitsiklis. The complexity of markov decision processes. *Mathematics of operations research*, 12(3):441–450, 1987.

H Van Dyke Parunak. Manufacturing experience with the contract net. *Distributed artificial intelligence*, 1:285–310, 1987.

Maarten Peeters, Katja Verbeeck, and Ann Nowé. Multi-agent learning in conflicting multi-level games with incomplete information. In *AAAI Technical Report (2)*, pages 73–80, 2004.

Bei Peng, Tabish Rashid, Christian Schroeder de Witt, Pierre-Alexandre Kamienny, Philip Torr, Wendelin Böhmer, and Shimon Whiteson. Facmac: Factored multi-agent centralised policy gradients. *Advances in Neural Information Processing Systems*, 34:12208–12221, 2021.

Leonid Peshkin, Kee-Eung Kim, Nicolas Meuleau, and Leslie Pack Kaelbling. Learning to cooperate via policy search. *arXiv preprint cs/0105032*, 2001.

Alexander Peysakhovich and Adam Lerer. Prosocial learning agents solve generalized stag hunts better than selfish ones. *arXiv preprint arXiv:1709.02865*, 2017.

Steven Pinker. The cognitive niche: Coevolution of intelligence, sociality, and language. *Proceedings of the National Academy of Sciences*, 107(supplement_2):8993–8999, 2010.

Jordan B Pollack, Alan D Blair, and Mark Land. Coevolution of a backgammon player. In *Artificial Life V: Proc. of the Fifth Int. Workshop on the Synthesis and Simulation of Living Systems*, pages 92–98. Cambridge, MA: The MIT Press, 1997.

Dean A Pomerleau. Efficient training of artificial neural networks for autonomous navigation. *Neural computation*, 3(1):88–97, 1991.

Rémy Portelas, Cédric Colas, Lilian Weng, Katja Hofmann, and Pierre-Yves Oudeyer. Automatic curriculum learning for deep rl: A short survey. In *IJCAI 2020-International Joint Conference on Artificial Intelligence*, 2021.

Mitchell A Potter. *The design and analysis of a computational model of cooperative coevolution*. George Mason University, 1997.

Mitchell A Potter and Kenneth A De Jong. A cooperative coevolutionary approach to function optimization. In *Parallel Problem Solving from Nature—PPSN III: International Conference on Evolutionary Computation The Third Conference on Parallel Problem Solving from Nature Jerusalem, Israel, October 9–14, 1994 Proceedings 3*, pages 249–257. Springer, 1994.

Mitchell A Potter, Lisa A Meeden, Alan C Schultz, et al. Heterogeneity in the coevolved behaviors of mobile robots: The emergence of specialists. In *International joint conference on artificial intelligence*, volume 17, pages 1337–1343. Citeseer, 2001.

David Premack and Guy Woodruff. Does the chimpanzee have a theory of mind? *Behavioral and brain sciences*, 1(4):515–526, 1978.

Arnu Pretorius, Scott Cameron, Andries Petrus Smit, Elan van Biljon, Lawrence Francis, Femi Azeez, Alexandre Laterre, and Karim Beguir. Learning to communicate through imagination with model-based deep multi-agent reinforcement learning, 2021. URL `https://openreview.net/forum?id=boZj4g3Jocj`.

Scott Proper and Kagan Tumer. Modeling difference rewards for multiagent learning. In *AAMAS*, pages 1397–1398, 2012.

Dean G Pruitt. *Negotiation behavior*. Academic Press, 1981.

Narendra Puppala, Sandip Sen, and Maria Gordin. Shared memory based cooperative coevolution. In *1998 IEEE International Conference on Evolutionary Computation Proceedings. IEEE World Congress on Computational Intelligence (Cat. No. 98TH8360)*, pages 570–574. IEEE, 1998.

Martin L Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

Neil Rabinowitz, Frank Perbet, Francis Song, Chiyuan Zhang, SM Ali Eslami, and Matthew Botvinick. Machine theory of mind. In *International conference on machine learning*, pages 4218–4227. PMLR, 2018.

Roberta Raileanu, Emily Denton, Arthur Szlam, and Rob Fergus. Modeling others using oneself in multi-agent reinforcement learning. In *International conference on machine learning*, pages 4257–4266. PMLR, 2018.

Tabish Rashid, Mikayel Samvelyan, Christian Schroeder Witt, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning. In *International Conference on Machine Learning*, pages 4292–4301, 2018.

Siddharth Reddy, Anca D. Dragan, and Sergey Levine. SQIL: Imitation learning via reinforcement learning with sparse rewards, 2019.

Cinjon Resnick, Wes Eldridge, David Ha, Denny Britz, Jakob Foerster, Julian Togelius, Kyunghyun Cho, and Joan Bruna. Pommerman: A multi-agent playground. *arXiv preprint arXiv:1809.07124*, 2018.

Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, pages 1530–1538, 2015.

Alan J Robinson and Lee Spector. Using genetic programming with multiple data types and automatic modularization to evolve decentralized and coordinated navigation in multi-agent systems. In *GECCO Late Breaking Papers*, pages 391–396, 2002.

Douglas LT Rohde and David C Plaut. Language acquisition in the absence of explicit negative evidence: How important is starting small? *Cognition*, 72(1):67–109, 1999.

Jeffrey S Rosenschein and Gilad Zlotkin. Designing conventions for automated negotiation. *AI magazine*, 15(3):29–29, 1994.

Jeffrey Solomon Rosenschein. Rational interaction: cooperation among intelligent agents. Technical report, Stanford Univ., CA (USA), 1986.

Christopher D Rosin and Richard K Belew. New methods for competitive coevolution. *Evolutionary computation*, 5(1):1–29, 1997.

Stéphane Ross and Drew Bagnell. Efficient reductions for imitation learning. In *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 661–668, 2010.

Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 627–635, 2011.

Julien Roy, Paul Barde, Félix Harvey, Derek Nowrouzezahrai, and Chris Pal. Promoting coordination through policy regularization in multi-agent deep reinforcement learning. *Advances in Neural Information Processing Systems*, 33:15774–15785, 2020.

Gavin A Rummery and Mahesan Niranjan. *On-line Q-learning using connectionist systems*, volume 37. University of Cambridge, Department of Engineering Cambridge, UK, 1994.

Rafal Salustowicz, Marco Wiering, and Juergen Schmidhuber. Learning team strategies with multiple policy-sharing agents: A soccer case study. In *ISDIA, Corso Elvezia 36, 6900*. Citeseer, 1997.

Rafał P Sałustowicz, Marco A Wiering, and Jürgen Schmidhuber. Learning team strategies: Soccer case studies. *Machine Learning*, 33(2):263–282, 1998.

Arthur L Samuel. Some studies in machine learning using the game of checkers. *IBM Journal of research and development*, 3(3):210–229, 1959.

Tuomas W Sandholm and Robert H Crites. On multiagent q-learning in a semi-competitive domain. In *Adaption and Learning in Multi-Agent Systems: IJCAI'95 Workshop Montréal, Canada, August 21, 1995 Proceedings 14*, pages 191–205. Springer, 1996.

Terence D Sanger. Neural network learning control of robot manipulators using gradually increasing task difficulty. *IEEE transactions on Robotics and Automation*, 10(3):323–333, 1994.

Hugo Santana, Geber Ramalho, Vincent Corruble, and Bohdana Ratitch. Multi-agent patrolling with reinforcement learning. In *Autonomous Agents and Multiagent Systems, International Joint Conference on*, volume 4, pages 1122–1129. IEEE Computer Society, 2004.

Fumihiro Sasaki, Tetsuya Yohira, and Atsuo Kawaguchi. Sample efficient imitation learning for continuous control. In *Proceedings of the 6th International Conference on Learning Representations (ICLR)*, 2018.

Arvind Sathi and Mark S Fox. Constraint-directed negotiation of resource reallocations. In *Distributed artificial intelligence*, pages 163–193. Elsevier, 1989.

Frederik Schadd, Sander Bakkes, and Pieter Spronck. Opponent modeling in real-time strategy games. In *GAMEON*, pages 61–70, 2007.

Jürgen Schmidhuber. Realistic multi-agent reinforcement learning. In *Learning in Distributed Artificial Intelligence Systems. Working Notes of the 1996 ECAI Workshop*. Citeseer, 1996.

Jürgen Schmidhuber and Jieyu Zhao. Multi-agent learning with the success-story algorithm. In *Distributed Artificial Intelligence Meets Machine Learning Learning in Multi-Agent Environments: ECAI'96 Workshop LDAIS Budapest, Hungary, August 13, 1996 ICMAS'96 Workshop LIOME Kyoto, Japan, December 10, 1996 Selected Papers*, pages 82–93. Springer, 2005.

Jeff Schneider, Weng-Keen Wong, Andrew Moore, and Martin Riedmiller. Distributed value functions. 1999.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, pages 1889–1897, 2015a.

John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015b.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

A Schultz, John J Grefenstette, and William Adams. Roboshepherd: Learning a complex behavior. *Robotics and manufacturing: Recent trends in research and applications*, 6: 763–768, 1996.

Peter Schuster and Karl Sigmund. Replicator dynamics. *Journal of theoretical biology*, 100 (3):533–538, 1983.

John R. Searle. *Collective Intentions and Actions*, page 90–105. Cambridge University Press, 2002. doi: 10.1017/CBO9780511606366.007.

Mahendra Sekaran and Sandip Sen. To help or not to help. In *Proceedings of the Seventeenth Annual Conference of the Cognitive Science Society*, pages 736–741. Citeseer, 1995.

Sandip Sen and Mahendra Sekaran. Using reciprocity to adapt to others. In *Adaption and Learning in Multi-Agent Systems: IJCAI'95 Workshop Montréal, Canada, August 21, 1995 Proceedings 14*, pages 206–217. Springer, 1996.

Sandip Sen and Mahendra Sekaran. Individual learning of coordination knowledge. *Journal of Experimental & Theoretical Artificial Intelligence*, 10(3):333–356, 1998.

Sandip Sen, Mahendra Sekaran, John Hale, et al. Learning to coordinate without sharing information. In *AAAI*, volume 94, pages 426–431, 1994.

Sandip Sen, Stephane Airiau, and Rajatish Mukherjee. Towards a pareto-optimal solution in general-sum games. In *Proceedings of the second international joint conference on Autonomous agents and multiagent systems*, pages 153–160, 2003.

Pier Giuseppe Sessa, Maryam Kamgarpour, and Andreas Krause. Efficient model-based multi-agent reinforcement learning via optimistic equilibrium computation. In *International Conference on Machine Learning*, pages 19580–19597. PMLR, 2022.

Tianyu Shi, Dong Chen, Kaian Chen, and Zhaojian Li. Offline reinforcement learning for autonomous driving with safety and exploration enhancement. *arXiv preprint arXiv:2110.07067*, 2021.

Yoav Shoham, Rob Powers, and Trond Grenager. On the agenda (s) of research on multi-agent learning. In *AAAI Technical Report (2)*, pages 89–95, 2004.

David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. Deterministic policy gradient algorithms. In *ICML*, 2014.

David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144, 2018.

George Gaylord Simpson. The baldwin effect. *Evolution*, 7(2):110–117, 1953.

Reid G Smith. The contract net protocol: High-level communication and control in a distributed problem solver. *IEEE Transactions on computers*, 29(12):1104–1113, 1980.

Kyunghwan Son, Daewoo Kim, Wan Ju Kang, David Earl Hostallero, and Yung Yi. Qtran: Learning to factorize with transformation for cooperative multi-agent reinforcement learning. In *International conference on machine learning*, pages 5887–5896. PMLR, 2019.

Jonathan Spencer, Sanjiban Choudhury, Arun Venkatraman, Brian Ziebart, and J Andrew Bagnell. Feedback in imitation learning: The three regimes of covariate shift. *arXiv preprint arXiv:2102.02872*, 2021.

Michael Spranger and Luc Steels. Co-acquisition of syntax and semantics-an investigation in spatial language. 2015.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

Randall Steeb, Stephanie Cammarata, Frederick A Hayes-Roth, Perry W Thorndyke, and Robert E Wesson. Distributed intelligence for air fleet control. Technical report, RAND CORP SANTA MONICA CA, 1981.

Luc Steels. A self-organizing spatial vocabulary. *Artificial life*, 2(3):319–332, 1995.

Luc Steels. Emergent adaptive lexicons. 1996.

Luc Steels. Modeling the cultural evolution of language. *Physics of life reviews*, 8(4):339–356, 2011.

Luc Steels and Manfred Hild. *Language grounding in robots*. Springer Science & Business Media, 2012.

Luc Steels and Frederic Kaplan. Collective learning and semiotic dynamics. In *Advances in Artificial Life: 5th European Conference, ECAL'99 Lausanne, Switzerland, September 13–17, 1999 Proceedings 5*, pages 679–688. Springer, 1999.

Luc Steels and Eörs Szathmáry. The evolutionary dynamics of language. *Biosystems*, 164: 128–137, 2018.

Kim Sterelny. *Thought in a Hostile World: The Evolution of Human Cognition*. Wiley-Blackwell, 2003.

Peter Stone and Manuela Veloso. Multiagent systems: A survey from a machine learning perspective. *Autonomous Robots*, 8:345–383, 2000.

Peter Stone, Gal Kaminka, Sarit Kraus, and Jeffrey Rosenschein. Ad hoc autonomous agent teams: Collaboration without pre-coordination. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 24, pages 1504–1509, 2010.

Peter Herald Stone. *Layered learning in multiagent systems*. Carnegie Mellon University, 1998.

Daniel Strouse, Max Kleiman-Weiner, Josh Tenenbaum, Matt Botvinick, and David J Schwab. Learning to share and hide intentions using information regularization. In *Advances in Neural Information Processing Systems*, pages 10270–10281, 2018.

Nobuo Suematsu and Akira Hayashi. A multiagent reinforcement learning algorithm using extended optimal response. In *Proceedings of the first international joint conference on Autonomous agents and multiagent systems: Part 1*, pages 370–377, 2002.

Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinicius Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z Leibo, Karl Tuyls, et al. Value-decomposition networks for cooperative multi-agent learning. *arXiv preprint arXiv:1706.05296*, 2017.

Dicky Suryadi and Piotr J Gmytrasiewicz. Learning models of other agents using influence diagrams. In *UM99 User Modeling: Proceedings of the Seventh International Conference*, pages 223–232. Springer, 1999.

Richard S Sutton. An adaptive network that constructs and uses and internal model of its world. *Cognition and Brain Theory*, 4(3):217–246, 1981.

Richard S Sutton. Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In *Machine learning proceedings 1990*, pages 216–224. Elsevier, 1990.

Richard S Sutton. Dyna, an integrated architecture for learning, planning, and reacting. *ACM Sigart Bulletin*, 2(4):160–163, 1991.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

Jonas Svennebring and Sven Koenig. Trail-laying robots for robust terrain coverage. In *2003 IEEE International Conference on Robotics and Automation (Cat. No. 03CH37422)*, volume 1, pages 75–82. IEEE, 2003.

John Sweller. Evolution of human cognitive architecture. *Psychology of learning and motivation*, 43:216–266, 2003.

Katia Sycara. Multiagent compromise via negotiation. In *Distributed artificial intelligence*, pages 119–137. Elsevier, 1989.

Milind Tambe. Recursive agent and agent-group tracking in a real-time dynamic environment. In *ICMAS*, pages 368–375, 1995.

Ming Tan. Multi-agent reinforcement learning: Independent vs. cooperative agents. In *Proceedings of the tenth international conference on machine learning*, pages 330–337, 1993.

Poj Tangamchit, John M Dolan, and Pradeep K Khosla. The necessity of average rewards in cooperative multirobot learning. In *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No. 02CH37292)*, volume 2, pages 1296–1301. IEEE, 2002.

Yong Meng Teo, Ba Linh Luong, and Claudia Szabo. Formalization of emergence in multi-agent systems. In *Proceedings of the 1st ACM SIGSIM Conference on Principles of Advanced Discrete Simulation*, pages 231–240, 2013.

Justin K Terry, Nathaniel Grammel, Ananth Hari, Luis Santos, and Benjamin Black. Revisiting parameter sharing in multi-agent deep reinforcement learning. *arXiv preprint arXiv:2005.13625*, 2020.

Gerald Tesauro. Practical issues in temporal difference learning. *Advances in neural information processing systems*, 4, 1991.

Gerald Tesauro and Jeffrey O Kephart. Pricing in agent economies using multi-agent q-learning. *Autonomous agents and multi-agent systems*, 5:289–304, 2002.

Wei-Cheng Tseng, Tsun-Hsuan Johnson Wang, Yen-Chen Lin, and Phillip Isola. Offline multi-agent reinforcement learning with knowledge distillation. *Advances in Neural Information Processing Systems*, 35:226–237, 2022.

Mycal Tucker, Yilun Zhou, and Julie Shah. Adversarially guided self-play for adopting social conventions. *arXiv preprint arXiv:2001.05994*, 2020.

Kagan Tumer and Adrian Agogino. Distributed agent-based air traffic flow management. In *Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems*, pages 1–8, 2007.

Kagan Tumer, Adrian K Agogino, and David H Wolpert. Learning sequences of actions in collectives of autonomous agents. In *Proceedings of the first international joint conference on Autonomous agents and multiagent systems: part 1*, pages 378–385, 2002.

Karl Tuyls, Katja Verbeeck, and Tom Lenaerts. A selection-mutation model for q-learning in multi-agent systems. In *Proceedings of the second international joint conference on Autonomous agents and multiagent systems*, pages 693–700, 2003.

George E Uhlenbeck and Leonard S Ornstein. On the theory of the brownian motion. *Physical review*, 36(5):823, 1930.

William Uther and Manuela Veloso. Adversarial reinforcement learning. Technical report, Tech. rep., Carnegie Mellon University. Unpublished, 1997.

H Van Dyke Parunak and Sven Brueckner. Entropy and self-organization in multi-agent systems. In *Proceedings of the fifth international conference on Autonomous agents*, pages 124–130, 2001.

LászlóZ Varga, Nick R Jennings, and David Cockburn. Integrating intelligent systems into a cooperating community for electricity distribution management. *Expert Systems with Applications*, 7(4):563–579, 1994.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Alexander Sasha Vezhnevets, Simon Osindero, Tom Schaul, Nicolas Heess, Max Jaderberg, David Silver, and Koray Kavukcuoglu. Feudal networks for hierarchical reinforcement learning. In *International Conference on Machine Learning*, pages 3540–3549. PMLR, 2017.

José M Vidal and Edmund H Durfee. Agents learning about agents: A framework and analysis. In *Collected papers from AAAI-97 workshop on Multiagent Learning*, pages 71–76, 1997.

Jose M Vidal and Edmund H Durfee. The moving target function problem in multi-agent learning. In *Proceedings International Conference on Multi Agent Systems (Cat. No. 98EX160)*, pages 317–324. IEEE, 1998.

Eugene Vinitsky, Nathan Lichtlé, Xiaomeng Yang, Brandon Amos, and Jakob Foerster. Nocturne: a scalable driving benchmark for bringing multi-agent learning one step closer to the real world. *arXiv preprint arXiv:2206.09889*, 2022.

Anna-Lisa Vollmer, Jonathan Grizou, Manuel Lopes, Katharina Rohlfing, and Pierre-Yves Oudeyer. Studying the co-construction of interaction protocols in collaborative tasks with humans. In *4th International Conference on Development and Learning and on Epigenetic Robotics*, pages 208–215. IEEE, 2014.

Anna-Lisa Vollmer, Britta Wrede, Katharina J Rohlfing, and Pierre-Yves Oudeyer. Pragmatic frames for teaching and learning in human–robot interaction: Review and challenges. *Frontiers in neurorobotics*, 10:10, 2016.

Lev S Vygotsky. *Thought and language*. MIT press, 1934.

Kyle Wagner. Cooperative strategies and the evolution of communication. *Artificial Life*, 6 (2):149–179, 2000.

Adam Walker and Michael J Wooldridge. Understanding the emergence of conventions in multi-agent systems. In *ICMAS*, volume 95, pages 384–389, 1995.

Jianhao Wang, Wenzhe Li, Haozhe Jiang, Guangxiang Zhu, Siyuan Li, and Chongjie Zhang. Offline reinforcement learning with reverse model-based imagination. *Advances in Neural Information Processing Systems*, 34:29420–29432, 2021.

Xiaofeng Wang and Tuomas Sandholm. Reinforcement learning to play an optimal nash equilibrium in team markov games. *Advances in neural information processing systems*, 15, 2002.

Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.

Richard A Watson and Jordan B Pollack. Coevolutionary dynamics in a minimal substrate. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2001)*, pages 702–709, 2001.

Robert Weihmayer and Hugo Velthuijsen. Application of distributed ai and cooperative problem solving to telecommunications, 1994.

Michael Weinberg and Jeffrey S Rosenschein. Best-response multiagent learning in non-stationary environments. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems-Volume 2*, pages 506–513, 2004.

Gerhard Weiss. *Multiagent systems: a modern approach to distributed artificial intelligence*. MIT press, 1999.

Greg Welch, Gary Bishop, et al. An introduction to the kalman filter. 1995.

Michael P Wellman and Junling Hu. Conjectural equilibrium in multiagent learning. *Machine Learning*, 33:179–200, 1998.

Keith J Werkman. Knowledge-based model of negotiation using shareable perspectives. In *Proceedings of the 10th Int Workshop on DAI*, 1990.

Keith J Werkman. Multiple agent cooperative design evaluation using negotiation. *Artificial Intelligence in Design'92*, pages 161–180, 1992.

R Paul Wiegand and Jayshree Sarma. Spatial embedding and loss of gradient in cooperative coevolutionary algorithms. In *Parallel Problem Solving from Nature-PPSN VIII: 8th International Conference, Birmingham, UK, September 18-22, 2004. Proceedings 8*, pages 912–921. Springer, 2004.

R Paul Wiegand, William C Liles, Kenneth A De Jong, et al. An empirical analysis of collaboration methods in cooperative coevolutionary algorithms. In *Proceedings of the genetic and evolutionary computation conference (GECCO)*, volume 2611, pages 1235–1245. Morgan Kaufmann San Francisco, 2001.

R Paul Wiegand, William C Liles, and Kenneth A De Jong. Analyzing cooperative coevolution with evolutionary game theory. In *Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No. 02TH8600)*, volume 2, pages 1600–1605. IEEE, 2002a.

R Paul Wiegand, William C Liles, and Kenneth A De Jong. Modeling variation in cooperative coevolution using evolutionary game theory. In *FOGA*, pages 203–220, 2002b.

Rudolf Paul Wiegand. *An analysis of cooperative coevolutionary algorithms*. George Mason University, 2004.

Marco Wiering, Rafał Sałustowicz, and Jürgen Schmidhuber. Reinforcement learning soccer teams with incomplete world models. *Autonomous Robots*, 7(1):77–88, 1999.

Daniël Willemsen, Mario Coppola, and Guido CHE de Croon. Mambpo: Sample-efficient multi-robot reinforcement learning using learned world models. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5635–5640. IEEE, 2021.

Andrew B Williams. Learning to share meaning in a multi-agent system. *Autonomous Agents and Multi-Agent Systems*, 8(2):165–193, 2004.

Robert C Wilson, Amitai Shenhav, Mark Straccia, and Jonathan D Cohen. The eighty five percent rule for optimal learning. *Nature communications*, 10(1):4646, 2019.

David H Wolpert and Kagan Tumer. Optimal payoff functions for members of collectives. *Advances in Complex Systems*, 4(02n03):265–279, 2001.

Mark Woodward, Chelsea Finn, and Karol Hausman. Learning to interactively learn and assist. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 2535–2543, 2020.

Michael Wooldridge, Stefan Bussmann, and Marcus Klosterberg. Production sequencing as negotiation. In *Proceedings of the First International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology (PAAM-96)*, pages 709–726. Citeseer, 1996.

Tengyang Xie, John Langford, Paul Mineiro, and Ida Momennejad. Interaction-grounded learning. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 11414–11423. PMLR, 18–24 Jul 2021. URL `https://proceedings.mlr.press/v139/xie21e.html`.

Holly Yanco and L Stein. An adaptive communication protocol for cooperating mobile robots. *From animals to animats*, 2:478–485, 1993.

Jiachen Yang, Ang Li, Mehrdad Farajtabar, Peter Sunehag, Edward Hughes, and Hongyuan Zha. Learning to incentivize other learning agents. *Advances in Neural Information Processing Systems*, 33:15208–15219, 2020.

Yiqin Yang, Xiaoteng Ma, Chenghao Li, Zewu Zheng, Qiyuan Zhang, Gao Huang, Jun Yang, and Qianchuan Zhao. Believe what you see: Implicit constraint approach for offline multi-agent reinforcement learning. *NeurIPS*, 2021.

Chao Yu, Akash Velu, Eugene Vinitsky, Jiaxuan Gao, Yu Wang, Alexandre Bayen, and Yi Wu. The surprising effectiveness of ppo in cooperative multi-agent games. *Advances in Neural Information Processing Systems*, 35:24611–24624, 2022.

Tianhe Yu, Garrett Thomas, Lantao Yu, Stefano Ermon, James Y Zou, Sergey Levine, Chelsea Finn, and Tengyu Ma. Mopo: Model-based offline policy optimization. *Advances in Neural Information Processing Systems*, 33:14129–14142, 2020.

Tianhe Yu, Aviral Kumar, Rafael Rafailov, Aravind Rajeswaran, Sergey Levine, and Chelsea Finn. Combo: Conservative offline model-based policy optimization. *Advances in neural information processing systems*, 34:28954–28967, 2021.

Byoung-Tak Zhang and Dong-Yeon Cho. Coevolutionary fitness switching: Learning complex collective behaviors using genetic programming. *Advances in genetic programming*, 3:425–445, 1999.

Chi Zhang, Sanmukh R Kuppannagari, Chuanxiu Xiong, Rajgopal Kannan, and Viktor K Prasanna. A cooperative multi-agent deep reinforcement learning framework for real-time residential load scheduling. In *Proceedings of the International Conference on Internet of Things Design and Implementation*, pages 59–69, 2019.

Chongjie Zhang and Victor Lesser. Multi-agent learning with policy prediction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 24, pages 927–934, 2010.

Chongjie Zhang and Victor Lesser. Coordinating multi-agent reinforcement learning with limited communication. In *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*, pages 1101–1108. International Foundation for Autonomous Agents and Multiagent Systems, 2013.

Qizhen Zhang, Chris Lu, Animesh Garg, and Jakob Foerster. Centralized model and exploration policy for multi-agent rl. In *Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems*, AAMAS '22, page 1500–1508, Richland, SC, 2022. International Foundation for Autonomous Agents and Multiagent Systems. ISBN 9781450392136.

Weinan Zhang, Xihuai Wang, Jian Shen, and Ming Zhou. Model-based multi-agent policy optimization with adaptive opponent-wise rollouts. *arXiv preprint arXiv:2105.03363*, 2021.

Jieyu Zhao and Jurgen Schmidhuber. Incremental self-improvement for life-time multi-agent reinforcement learning. In *From Animals to Animats 4: Proceedings of the Fourth International Conference on Simulation of Adaptive Behavior, Cambridge, MA*, pages 516–525. MIT Press, Bradford Books Cambridge, MA, 1996.

Lianmin Zheng, Jiacheng Yang, Han Cai, Ming Zhou, Weinan Zhang, Jun Wang, and Yong Yu. Magent: A many-agent reinforcement learning platform for artificial collective intelligence. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

Hongwei Zhou, Yichen Gong, Luvneesh Mugrai, Ahmed Khalifa, Andy Nealen, and Julian Togelius. A hybrid search agent in pommerman. In *Proceedings of the 13th International Conference on the Foundations of Digital Games (FDG)*, pages 1–4, 2018.

Brian D Ziebart. *Modeling purposeful adaptive behavior with the principle of maximum causal entropy*. PhD thesis, 2010.

Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, and Anind K Dey. Maximum entropy inverse reinforcement learning. In *Proceedings of the 23rd AAAI Conference on Artificial Intelligence*, pages 1433–1438, 2008.

Gilad Zlotkin and Jeffrey S Rosenschein. Blocks, lies, and postal freight: The nature of deception in behaviour. In *Proc of the 10th Int Workshop on DAI*, 1990.