Towards Efficient Cooperation within Learning Agents

Rundong Wang

School of Computer Science and Engineering

A thesis submitted to the Nanyang Technological University in partial fulfillment of the requirements for the degree of Doctor of Philosophy

2023

Statement of Originality

I hereby certify that the work embodied in this thesis is the result of original research, is free of plagiarised materials, and has not been submitted for a higher degree to any other University or Institution.

14/04/2023

Date

Rundong Wang

Rundong Wang

Supervisor Declaration Statement

I have reviewed the content and presentation style of this thesis and declare it is free of plagiarism and of sufficient grammatical clarity to be examined. To the best of my knowledge, the research and writing are those of the candidate except as acknowledged in the Author Attribution Statement. I confirm that the investigations were conducted in accord with the ethics policies and integrity standards of Nanyang Technological University and that the research data are presented honestly and without prejudice.

14/04/2023	[illegible]
....	....
Date	Prof. Zinovi Rabinovich

Authorship Attribution Statement

This thesis contains material from 3 paper(s) published in the following peer-reviewed journal(s) / from papers accepted at conferences, and 1 paper under review, in which I am listed as an author.

Chapter 3 is published as Rundong Wang, Runsheng Yu, Bo An, Zinovi Rabinovich. I^{2}HRL: Interactive Influence-based Hierarchical Reinforcement Learning. *International Joint Conference on Artificial Intelligence*, 2020.

The contributions of the co-authors are as follows:

- I proposed the initial project direction, prepared the manuscript drafts, and performed all the experimental work.
- The manuscript was revised together by Runsheng and Prof. An.
- A/Prof. Rabinovich discussed with me the details of the implementation.

Chapter 4 is published as Rundong Wang, Xu He, Runsheng Yu, Wei Qiu, Bo An, Zinovi Rabinovich. Learning Efficient Multi-agent Communication: An Information Bottleneck Approach. , *International Conference on Machine Learning*, 2020.

The contributions of the co-authors are as follows:

- I proposed the initial project direction, prepared the manuscript drafts, and performed all the experimental work.
- The manuscript was revised together by A/Prof. Rabinovich and Prof. An.
- Xu He, Runsheng Yu, and Wei Qiu discussed with me the details of the implementation.

Chapter 5 is published as Rundong Wang, Hongxin Wei, Bo An, Zhouyan Feng, Jun Yao. Commission Fee is not Enough: A Hierarchical Reinforced Framework for Portfolio Management". *The Thirty-Fifth AAAI Conference on Artificial Intelligence*, 2021.

The contributions of the co-authors are as follows:

- Prof. An proposed the initial project direction
- I prepared the manuscript drafts, and performed all the experimental work.
- The manuscript was revised together by Jun Yao and Prof. An.
- Hongxin Wei and Zhouyan Feng discussed with me the details of the implementation.

Chapter 6 is under review, and can be access in Arxiv as Rundong Wang, Longtao Zheng, Wei Qiu, Bowei He, Bo An, Zinovi Rabinovich, Yujing Hu, Yingfeng Chen, Tangjie Lv, Changjie Fan. Towards Skill and Population Curriculum for MARL. 2022.

The contributions of the co-authors are as follows:

- Yujing Hu proposed the initial project direction, and Yingfeng Chen, Tangjie Lv, and Changjie Fan provided the computing resources.
- I prepared the manuscript drafts. Longtao Zheng and me performed all the experimental work.
- The manuscript was revised together by A/Prof. Rabinovich and Prof. An.
- Longtao Zheng, Wei Qiu, and Bowei He discussed with me the details of the implementation.

14/04/2023 [illegible]

. . . .

Date Rundong Wang

Acknowledgements

First and foremost, I am greatly indebted to my supervisors, Prof. Zinovi Rabinovich and Prof. Bo An, who, throughout my four-year journey at Nanyang Technological University, have provided me insightful discussions, a suitable research environment, as well as the freedom that allows me to explore interesting research problems. Their great sense of research taste encourages and inspires me to work on important questions. Not only did I learn about how to be professional, but also I learn from his methods and thoughts to tackle research problems. I could not have imagined having a better advisor and mentor for my Ph.D. study.

I am also very fortunate to meet and learn from a group of talented people: Qingyu Guo, Wanyuan Wang, Haipeng Chen, Yanhai Xiong, Mengchen Zhao, Jiuchuan Jiang, Youzhi Zhang, Xinrun Wang, Lei Feng, Aye Phyu, Hongxin Wei, Xu He, Wei Qiu, Hang Xu, Jakub Cerny, Runsheng Yu, Yanchen Deng, Shuxin Li, Wanqi Xue, Fei Fei Lin, Zhuyun Yin, Xinyu Cai, Shuo Sun, Longtao Zheng, Pengdeng Li, and Shufeng Kong. I would like to all of them for the joy, the support, the excitement that they gave me.

I would like to thank my Thesis Advisory Committee (TAC) members: Prof. Gao Cong and Prof. Xiaohui Bei. They helped me a lot and made my research journey easier at NTU.

During my Ph.D., I had an exciting internship at Tencent AI Lab and Sea AI Lab. I would like to thank Weixuan Wang, Xianhan Zeng, Zhenjie Lian, Yiming Gao, Feiyu Liu, Siqin Li, Xianliang Wang, Lanxiao Huang, Liang Wang, Qiang Fu, Yang Wei, Pengqian Yu, Zhongwen Xu, Prof. Shuicheng Yan and other people I met at Shenzhen and Singapore, who helped me a lot during the internship.

I would like to thank many of my friends: Zhuoyi Lin, Peng Zhao, Chong Wu, Jiang Zhu, Xiuyang Xia and so on. The bond built among us is one of the most cherished things in my life.

Most importantly, I would like to thank my love, Mengyu Huang and families, especially my parents: Haiyan Duan and Peibin Wang, for their unconditional love and support to me. Needless to say, this thesis would not have been possible without your encouragement along the way. This thesis is dedicated to all of you. I love you.

Abstract

A wide range of real world problems, such as the control of autonomous vehicles and drones, packet delivery, and many others consists of a number of agents that need to take actions based on local observations and can thus be formulated in the multi-agent reinforcement learning (MARL) setting. Furthermore, as more machine learning systems are deployed in the real world, they will start having impact on each other, effectively turning most decision making problems into multi-agent cooperation problems. In this doctoral thesis, we develop and evaluate novel deep RL (DRL) methods that address the unique challenges which arise in these settings. These challenges include learning to communicate, to collaborate, and to reciprocate amongst agents.

In the first second part of the doctoral thesis, we consider the problem of the limited-bandwidth communication for multi-agent reinforcement learning, where agents cooperate with the assistance of a communication protocol. A key difficulty, faced by a group of learning agents in real-world domains, is the need to efficiently exploit the available communication resources, such as limited bandwidth. To address the limited bandwidth problem, we develop an *Informative Multi-Agent Communication* (IMAC) method to learn efficient communication protocols by compressing the communication messages. From the perspective of communication theory, we prove that the limited bandwidth constraint requires low-entropy messages throughout the transmission. In IMAC, inspired by the information bottleneck principle, agents are trained to learn a valuable and compact communication protocol.

The second part of the doctoral thesis investigates the challenges in hierarchical reinforcement learning (HRL), which is often implemented as a high-level policy assigning subgoals to a low-level policy. HRL suffers the high-level non-stationarity problem since the low-level policy is constantly changing. The non-stationarity also leads to the data efficiency problem: policies need more data at non-stationary states to stabilize training. To address these issues, we propose a novel HRL method: *Interactive Influence-based Hierarchical Reinforcement Learning* (I^{2}HRL). In I^{2}HRL, we enable the interaction between the low-level and high-level policies, i.e., the low-level policy sends its policy representation to the high-level policy. The key insight here is that "hierarchy" is just a way to assign responsibilities in a complex system, so HRL is actually about "collaboration" among multiple agents and can be interpreted as a form of MARL when we consider each level policy as agent. The state transition function and the reward function of each agent depend on the actions of all agents. Besides, in this part, we consider the specific problem of Fintech: portfolio management via reinforcement learning, which explores how to optimally reallocate a fund into different financial assets over the long term by trial-and-error. Existing methods are impractical since they usually assume each reallocation can be finished immediately and thus ignoring the price slippage as part of the trading cost. To address these issues, we propose a hierarchical reinforced stock trading system for portfolio management (*HRPM*). Concretely, we decompose the trading process into a hierarchy of portfolio management over trade execution and train the corresponding policies. The high-level policy gives portfolio weights at a lower frequency to maximize the long term profit and invokes the low-level policy to sell or buy the corresponding shares within a short time window at a higher frequency to minimize the trading cost. We train two levels of policies via pre-training scheme and iterative training scheme for data efficiency. Extensive experimental results in the U.S. market and the China market demonstrate that *HRPM* achieves significant improvement against many state-of-the-art approaches.

In the third part of the doctoral thesis, we consider the problem of automatic curriculum learning in MARL, where a teacher and a student will learn from each other and reciprocate each other. Recent advances in multi-agent reinforcement learning (MARL) allow agents to coordinate their behaviors in complex environments. However, common MARL algorithms still suffer from scalability and sparse reward issues. One promising approach to resolving them is *automatic curriculum learning* (ACL). ACL involves a *student* (curriculum learner) training on tasks of increasing difficulty controlled by a *teacher* (curriculum generator). Despite its success, ACL's applicability is limited by (1) the lack of a general student framework for dealing with the varying number of agents across tasks and the sparse reward problem, and (2) the non-stationarity of the teacher's task due to ever-changing student strategies. As a remedy for ACL, we introduce a novel automatic curriculum learning framework, Skilled Population Curriculum (SPC), which adapts curriculum learning to multi-agent coordination. Specifically, we endow the student with population-invariant communication and a hierarchical skill set, allowing it to learn cooperation and behavior skills from distinct tasks with varying numbers of agents. In addition, we model the teacher as a contextual bandit conditioned by student policies, enabling a team of agents to change its size while still retaining previously acquired skills. We also analyze the inherent non-stationarity of this multi-agent automatic curriculum teaching problem and provide a corresponding regret bound. Empirical results show that our method improves the performance, scalability and sample efficiency in several MARL environments.

To conclude, this thesis makes progress on the challenges that arise in hierarchical and multi-agent settings and also opens-up a number of exciting questions for future research. These include how agents can learn to account for the learning of other agents when their rewards or observations are unknown, how to learn communication protocols in settings of partial common interest, and how to account for the agency of humans in the environment.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Deep Reinforcement Learning

Deep Reinforcement Learning (DRL) has seen significant progress in recent years in the development of learning algorithms that can solve static tasks, where a single agent operates in an environment. However, the real world is full of multi-agent environments where a large number of distributed agents must take independent decisions based on local observations to achieve a shared goal or maximize individual rewards. Examples of such environments include self-driving cars, robotic assembly lines, and multi-player games. In these environments, not only do agents interact with each other, but they must also consider the presence of other biological agents such as humans or animals.

To address these challenges, Multi-Agent Reinforcement Learning (MARL) has emerged as a field of study that aims to develop and analyze learning algorithms that can discover effective policies in multi-agent settings. Furthermore, as AI systems become more influential and interact with humans, they are likely to affect the personality development of the human users themselves. For example, AI systems that serve user preferences should not assume that the user's preferences are static. Instead, they need to consider the agency of users by reasoning over their beliefs, ideals, and desires. MARL can help develop AI systems that can interact with humans in a more empathetic and efficient way, improving the user experience and avoiding unintended negative consequences.

In addition, MARL can serve as a stepping stone towards developing systems that have human-like reasoning abilities. The development of human-level intelligence occurred in the social context of many agents interacting, and research suggests that higher level cognitive skills such as abstraction could naturally arise in multi-agent settings. [2–6] This is because agents must account for the current state of mind of other agents as part of their state of mind, which corresponds to a level of abstract thought. By studying and developing MARL algorithms, we can gain insights into how to create intelligent systems that can reason about the beliefs, desires, and perspectives of other agents, and thereby make decisions that are aligned with the goals of the system.

Despite its potential, current MARL approaches face several limitations, including non-stationary environments, sample inefficiency, and the curse of dimensionality. These limitations have inspired the studies in this thesis, which seek to address these shortcomings and contribute to the advancement of MARL. Non-stationary environments pose a significant challenge in MARL, as the behavior of other agents can change over time, making it difficult for an agent to learn an optimal policy. Non-stationarity implies the inability of individual agents to attribute received rewards to the actions of others, rather than their own actions – because they lack the necessary information. Communication has the ability to correct this lack of information but adds an additional layer of decision-making. This thesis presents novel algorithms that enable agents to adapt to the changing behavior of other agents, thus improving their performance in non-stationary environments. Moreover, sample inefficiency is another limitation of MARL, as agents often require a large number of interactions with the environment to learn effective policies. This thesis introduces new techniques that improve sample efficiency by leveraging the structure of the problem and the relationships between agents, allowing them to learn more effectively from fewer interactions. Lastly, the curse of dimensionality is a well-known issue in reinforcement learning, and it becomes even more pronounced in MARL due to the increased complexity of multi-agent environments. This thesis addresses this challenge by developing hierarchical and modular approaches that can break down complex tasks into simpler sub-tasks, reducing the dimensionality of the problem and enabling more efficient learning. To sum up, the contributions of this thesis are aimed at addressing the key shortcomings of existing MARL approaches, including non-stationary environments, sample inefficiency, and the curse of dimensionality. Specifically, the thesis presents novel algorithms

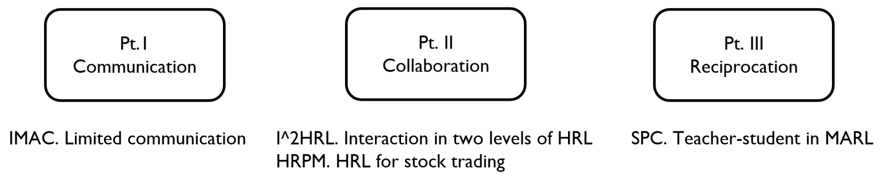

Pt. I
Communication
Pt. II
Collaboration
Pt. III
Reciprocation
IMAC. Limited communication
I^2HRL. Interaction in two levels of HRL
HRPM. HRL for stock trading
SPC. Teacher-student in MARL

FIGURE 1.1: The overall perspective of this thesis

.

and techniques that enhance the efficiency of multi-agent communication, improve the ability of agents to collaborate in hierarchical settings, and facilitate better reciprocation among agents in various application domains. By addressing these limitations, the thesis aims to advance the state-of-the-art in MARL and contribute to the development of more sophisticated and effective multi-agent systems that can tackle complex real-world challenges.

1.2 Organization of the thesis

This thesis is organized into three parts, each focusing on a different aspect of multi-agent systems.

Chapter 1 introduces the multi-agent RL setting and provides the necessary algorithmic and conceptual tools of deep reinforcement learning. Chapter 2 provides the preliminary knowledge about RL and related work of this thesis.

Part I of the thesis focuses on communication, specifically developing an informative multi-agent communication method in a limited bandwidth environment. In Chapter 4, we address the issue of efficient communication between agents in limited bandwidth environments. We introduce the Informative Multi-Agent Communication (IMAC) method, which learns communication protocols and scheduling while considering the limited bandwidth constraint. The method is based on the information bottleneck principle and employs a customized batch-norm layer to control message entropy. Through extensive experiments, we demonstrate that IMAC leads to faster convergence and more efficient communication among agents under limited bandwidth compared to other baseline methods.

Part II of the thesis concentrates on collaboration. In Chapter 3, we propose a novel Hierarchical Reinforcement Learning (HRL) algorithm called Interactive Influence-based HRL (I^2HRL). By introducing interaction between the low-level and high-level policies, we enable the policies to collaborate and learn from each other. Our approach significantly improves the learning performance and acceleration compared to state-of-the-art HRL methods in various tasks.

Chapter 5 focuses on the problem of portfolio management with trading cost via deep reinforcement learning. We introduce a hierarchical reinforced stock trading system (HRPM) that employs a hierarchy of portfolio management over trade execution and trains corresponding policies. The high-level policy determines portfolio weights, and the low-level policy executes trades. Extensive experiments show that HRPM outperforms many state-of-the-art approaches in the US and China markets.

Part III of the thesis focuses on reciprocation, specifically addressing the scalability and sparse reward issues in multi-agent systems. In Chapter 6, we address scalability and sparse reward issues in multi-agent systems. We introduce the Curriculum Oriented Skills and Tactics (COST) algorithm, which uses a population-invariant multi-agent communication framework to handle varying numbers of agents and a hierarchical scheme for agents to learn skills to deal with sparse rewards. To mitigate non-stationarity, we model the teacher as a contextual bandit. Our experiments show that COST achieves state-of-the-art performance in several tasks in the multi-particle environment and the challenging 5vs5 competition in GRF.

Chapter 7 concludes the report and suggests ideas for future work. The thesis contributes to the development of communication, collaboration, and reciprocation between agents in various settings, demonstrating the effectiveness of the proposed algorithms in improving the efficiency of interaction and collaboration among agents.

Chapter 2

Preliminary

2.1 Markov Decision Process

The Markov Decision Process (MDP) is a well-established mathematical framework used in the field of reinforcement learning to model decision-making problems involving an agent interacting with an environment. It is defined by a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$, where each component has a specific meaning.

- The set of states, denoted as $\mathcal{S}$, represents the possible situations or configurations of the environment. States are used to capture the relevant information about the environment that the agent needs to make decisions.
- The set of actions, denoted as $\mathcal{A}$, represents the possible choices or decisions that the agent can make. Actions are taken by the agent to transition from one state to another in the environment.
- The state transition probability function, denoted as $\mathcal{P}$, represents the probability of transitioning from one state to another after taking a certain action. It is usually expressed as $\mathcal{P}(s'|s, a)$, where s' is the next state, s is the current state, and a is the action taken. This function captures the dynamics of the environment and describes how the environment changes in response to the agent's actions.
- The reward function, denoted as $\mathcal{R}$, represents the immediate reward that the agent receives after taking a certain action in a certain state. It is usually

expressed as $\mathcal{R}(s, a, s')$, where s is the current state, a is the action taken, and s' is the next state. The reward function provides feedback to the agent about the desirability of its actions and serves as a guiding signal for learning.

- The discount factor, denoted as γ, is a scalar value between 0 and 1 that represents the preference of the agent for immediate rewards versus future rewards. A discount factor of 0 indicates that the agent only cares about immediate rewards, while a discount factor of 1 indicates that the agent values future rewards equally to immediate rewards. The discount factor allows the agent to trade-off between short-term and long-term rewards in its decision-making process.

The agent selects actions based on a policy, denoted as $\pi(a \mid s)$, which represents the probability of selecting action a given the current state s. The goal of the agent is to learn an optimal policy, denoted as π^*, that maximizes the expected cumulative reward or the expected sum of discounted rewards over an infinite time horizon, which is defined as $G_t = \sum_{k=0}^{\infty} \gamma^k r_{k+t}$. This is typically achieved through reinforcement learning algorithms that update the policy and value functions based on the observed interactions with the environment, allowing the agent to learn from experience and make informed decisions.

The MDP framework provides a formal and well-defined framework for studying and solving decision-making problems in the field of reinforcement learning, with applications in various domains, such as robotics, game playing, finance, and healthcare. Overall, MDPs are a fundamental tool for understanding and solving sequential decision-making problems in artificial intelligence and machine learning. With their clear mathematical representation, MDPs provide a solid foundation for developing and analyzing decision-making algorithms for autonomous agents.

2.2 Reinforcement Learning

Reinforcement learning (RL), a widely used framework for solving problems that can be modeled as MDPs, relies on the concept of policy and value function. The policy dictates the actions to be taken at each state or time step, while the value function quantifies the desirability of states or state-action pairs. These functions can be represented using tables or approximated using functions, and are essential for decision making and learning in reinforcement learning.

In this section, we first introduce the state value function, the state-action value function, and the Bellman equation. We then prove the convergence of iteration methods using the Banach fixed-point theorem. For a given policy, we can compute its state value function by repeatedly applying the Bellman equation. Each iteration can be considered as a mapping from the space of value functions to itself, and the Bellman operator $\mathcal{T}_\pi$ is a contraction mapping. Thus, by the Banach fixed-point theorem, the iteration method converges to a unique fixed point, which is the true value function under the policy. Lastly, we present several value-based methods, such as Q-learning and SARSA, which directly estimate the state-action value function and then derive the policy from it. We also introduce policy-based methods, such as REINFORCE and actor-critic, which directly parameterize the policy and optimize it using gradient-based methods.

The state value function, denoted as $V_\pi(s)$, represents the expected cumulative discounted reward, or the long-term reward, starting from a given state s and following a policy π to choose actions. It is defined as the expected value of the sum of discounted rewards from the current time step t onwards, given that the agent is at state $s_t = s$ and follows policy π:

$$V_\pi(s) = \mathbb{E}\pi \left[G_t \mid s_t = s \right] = \mathbb{E}\pi \left[\sum_{k=0}^{\infty} \gamma^k r_{k+t} \mid s_t = s \right], \tag{2.1}$$

where γ is a discount factor that determines the weight of future rewards, and r_{k+t} is the reward obtained at time step $k+t$. The objective of reinforcement learning is to find the optimal policy π that maximizes the expected cumulative reward starting from the initial state, denoted by $\mathcal{J}(\pi) = \mathbb{E}s_0 \sim \rho_0[V\pi(s_0)]$, where ρ_0 is the initial distribution of states.

Similarly, the state-action value function, denoted as $Q_\pi(s, a)$, represents the expected cumulative discounted reward, or the long-term reward, starting from a given state s, taking action a, and following policy π thereafter. It is defined as the expected value of the sum of discounted rewards from the current time step t onwards, given that the agent is at state $s_t = s$ and takes action $a_t = a$:

$$Q_\pi(s,a) = \mathbb{E}\pi\left[G_t \mid s_t = s, a_t = a\right] = \mathbb{E}\pi\left[\sum_{k=0}^{\infty}\gamma^k r_{k+t} \mid s_t = s, a_t = a\right]. \tag{2.2}$$

The Bellman equations, as shown above, express the relationship between the value of current and successor states or state-action pairs. These equations are the foundation of the policy iteration method, a widely used approach for solving MDPs, which consists of two main steps: policy evaluation and policy improvement. By iteratively executing these two steps, the optimal policy can be obtained in tabular cases. Accurate estimation of the policy and value functions is crucial for successful reinforcement learning.

According to the above two definitions, some properties of these two value functions can be given:

$$\begin{aligned} Q_\pi\left(s_t, a_t\right) &= \mathbb{E}_{s_{t+1}\sim\mathcal{P}(s_t,a_t)}\left[r_{t+1} + \gamma V\left(s_{t+1}\right)\right], \\ V_\pi\left(s_t\right) &= \mathbb{E}_{s_{t+1}\sim\mathcal{P}(s_t,a_t),a_t\sim\pi(s_t)}\left[r_{t+1} + \gamma V\left(s_{t+1}\right)\right], \\ Q_\pi\left(s_t, a_t\right) &= \mathbb{E}_{s_{t+1}\sim\mathcal{P}(s_t,a_t),a_{t+1}\sim\pi(s_{t+1})}\left[r_{t+1} + \gamma Q\left(s_{t+1}, a_{t+1}\right)\right], \end{aligned} \tag{2.3}$$

which are Bellman equations that express the relationship between the value of current and successor states or state-action pairs.

Based on Bellman equations, the policy iteration and the value iteration method are proposed to solve MDPs. Before presenting these methods, we first introduce the Banach fixed-point theorem, which supports the convergence of iteration methods.

Theorem 2.1 (Banach fixed-point theorem)**.** *Let X be a complete metric space, and f be a contraction on X. Then there exists a unique x^* such that $f\left(x^*\right) = x^*$. The contraction of a mapping is defined as follows.*

The Banach fixed-point theorem provides a powerful tool for ensuring the convergence of iterative methods in various settings. In particular, the theorem guarantees the existence and uniqueness of a fixed point for a contraction mapping on a complete metric space. In the context of RL, the metric space is typically taken to be the set of real numbers $\mathcal{R}$, and the norm is the absolute value function.

Definition 2.1 (Contraction)**.** Let X be a metric space, and $f : X \longrightarrow X$. We will say that f is a contraction if there exists some $0 < k < 1$ such that $d(f(x), f(y)) \geq kd(x, y)$ for all $x, y \in X$. The inf of such k 's is called the contraction coefficient.

The Banach fixed-point theorem provides a powerful tool for ensuring the convergence of iterative methods in various settings. In particular, the theorem guarantees the existence and uniqueness of a fixed point for a contraction mapping on a complete metric space. In the context of RL, the metric space is typically taken to be the set of real numbers $\mathcal{R}$, and the norm is the absolute value function, i.e., X usually is $\mathcal{R}$ and d is the norm. Then we can prove that the Bellman equation has a unique fixed point as follows:

$$\begin{aligned}
\left\|\mathcal{T}^\pi V_1 - \mathcal{T}^\pi V_2\right\|_\infty &= \left\|r + \gamma \mathcal{P}_\pi V_1 - r - \gamma \mathcal{P}_\pi V_2\right\|_\infty \\
&= \gamma \left\|\mathcal{P}_\pi \left(V_1 - V_2\right)\right\|_\infty \\
&\leqslant \gamma \left\|V_1 - V_2\right\|_\infty ,
\end{aligned} \tag{2.4}$$

where $\mathcal{T}^\pi$ is the Bellman operator representing the left side of the Bellman equation and $\mathcal{P}_\pi$ is the transition probabilities following π.

Policy iteration includes two parts: policy evaluation and policy improvement. By alternatively executing these two parts, we can obtain the optimal policy in tabular cases.

The policy evaluation step involves computing the state-value function V_π for a given policy π using the Bellman function. This can be done by iteratively updating the value estimates for each state until convergence is reached. Specifically, for each state s_t, the value estimate is updated as:

$$\begin{aligned}
V_\pi\left(s_t\right) &= \mathbb{E}_{s_{t+1} \sim \mathcal{P}(s_t, a_t), a_t \sim \pi(s_t)}\left[r_{t+1} + \gamma V\left(s_{t+1}\right)\right] \\
&= \sum_a \pi(a \mid s) \sum_{s', r} \mathcal{P}\left(s', r \mid s, a\right)\left[r + \gamma V\left(s'\right)\right],
\end{aligned} \tag{2.5}$$

where $\mathcal{P}$ is the transition probability function, and r_t is the reward obtained after taking action a_t in state s_t.

If the transition probability function $\mathcal{P}$ is known, we can repeatedly update $V(s)$ by $V\left(s'\right)$ for all the $s \in S$, until the gap between the new and old values is small enough. According to the Banach fixed-point theorem, this iterative process converges to the unique fixed-point solution V_π.

Once we have the value function V_π for a policy π, we can improve the policy using the policy improvement theorem. Specifically, we can compute the action-value function $Q_\pi(s, a)$ for each state-action pair, and update the policy at each state s by selecting the action that maximizes $Q_\pi(s, a)$, i.e.,

$$\pi'(s) = \arg\max_a Q_\pi(s, a) \tag{2.6}$$

The policy improvement theorem states that the new policy π' obtained in this way is guaranteed to be at least as good as the original policy π, i.e., $V_{\pi'}(s) \geq V_\pi(s)$ for all states s. By iteratively performing policy evaluation and improvement, we can converge to the optimal policy π^* that maximizes the expected return $\mathcal{J}(\pi)$.

Theorem 2.2 (Policy improvement theorem)**.** *When $\pi'(s) = \arg\max_a Q_\pi(s, a)$, the return of π' will be no less than that of π :*

$$V_{\pi'}(s) \leq V_\pi(s) \tag{2.7}$$

Proof.

$$\begin{aligned} V_\pi(s) &\leq Q_\pi\left(s, \pi'(s)\right) \\ &= \mathbb{E}_{\pi'}\left[r_t + \gamma V_\pi\left(s_{t+1}\right) \mid s_t\right] \\ &\leq \mathbb{E}_{\pi'}\left[r_t + \gamma Q_\pi\left(s_{t+1}, \pi'\left(s_{t+1}\right)\right) \mid s_t\right] \\ &= \mathbb{E}_{\pi'}\left[r_t + \gamma \mathbb{E}_{\pi'}\left[r_{t+1} + \gamma V_\pi\left(s_{t+2}\right)\right] \mid s_t\right] \\ &= \mathbb{E}_{\pi'}\left[r_t + \gamma r_{t+1} + \gamma^2 V_\pi\left(s_{t+2}\right) \mid s_t\right] \\ &\leq \mathbb{E}_{\pi'}\left[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \gamma^3 V_\pi\left(s_{t+3}\right) \mid s_t\right] \\ &\vdots \\ &\leq \mathbb{E}_{\pi'}\left[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \gamma^3 r_{t+3} + \cdots \mid s_t\right] \\ &= V_{\pi'}(s). \end{aligned} \tag{2.8}$$

If V_π is the same as $V_{\pi'}$, we find the fixed point of the Bellman equation. According to the fixed-point theorem, both π and π' must be the optimal policies. □

Value iteration. Policy evaluation usually requires multiple sweeps through the state set to obtain accurate values, which is time-consuming. Value iteration simplifies the policy iteration method by restricting the number of sweeps to one.

Formally,

$$V_{\pi}\left(s_t\right)=\max_a \sum_{s^{\prime}, r} \mathcal{P}\left(s^{\prime}, r \mid s, a\right)\left[r+\gamma V\left(s^{\prime}\right)\right], \tag{2.9}$$

which does not lose the convergence guarantees of policy iteration.

2.2.1 Value-based methods

In the above-mentioned policy iteration or value iteration methods, they both need to obtain the expectation during the update. Thus, Temporal-Difference (TD) learning is proposed to estimate the expectation of V values. To estimate V values of a policy π, let π interact with the environment with T time steps. Then, we can update V values using sampled data by

$$V\left(s_t\right)=V\left(s_t\right)+\alpha\left[r_t+V\left(s_{t+1}\right)-V\left(s_t\right)\right],$$

where α is a constant to decide the weight of new samples.

SARSA [7]

SARSA stands for State-Action-Reward-State-Action, and it is an on-policy algorithm that learns the optimal Q-function through experience. At each time step, SARSA updates the Q-value of the current state-action pair based on the reward received and the Q-value of the next state-action pair, which is chosen based on the current policy. The update rule for SARSA is:

$$Q\left(s_t, a_t\right) \leftarrow Q\left(s_t, a_t\right)+\alpha\left[r_t+\gamma Q\left(s_{t+1}, a_{t+1}\right)-Q\left(s_t, a_t\right)\right],$$

where $Q(s_t, a_t)$ is the Q-value of state-action pair (s_t, a_t), r_t is the reward received at time t, α is the learning rate, γ is the discount factor, and a_{t+1} is the action selected for the next state s_{t+1} using the current policy. The policy can be an ϵ-greedy policy that selects the action with the highest Q-value with probability $1-\epsilon$ and a random action with probability ϵ.

Q-learning [8]

Q-learning, on the other hand, is an off-policy algorithm that learns the optimal Q-function by selecting the action that maximizes the Q-value of the next state-action pair. At each time step, Q-learning updates the Q-value of the current state-action pair based on the reward received and the maximum Q-value of the next state-action pair, which is chosen greedily. The update rule for Q-learning is:

$$Q\left(s_t, a_t\right) \leftarrow Q\left(s_t, a_t\right) + \alpha\left[r_t + \gamma \max_{a'} Q\left(s_{t+1}, a'\right) - Q\left(s_t, a_t\right)\right],$$

where $Q(s_t, a_t)$ is the Q-value of state-action pair (s_t, a_t), r_t is the reward received at time t, α is the learning rate, γ is the discount factor, and $\max_{a'} Q(s_{t+1}, a')$ is the maximum Q-value of the next state-action pair. The policy used for selecting actions during the learning process can be a greedy policy that selects the action with the highest Q-value. However, during the evaluation phase, a different policy can be used, such as an ϵ-greedy policy, to balance exploration and exploitation.

Deep Q-Learning (DQL) [9]

DQL is a popular reinforcement learning method for solving large-scale problems with high-dimensional state spaces. DQL employs a deep neural network to approximate the Q-value function, which represents the expected discounted sum of rewards that an agent can receive by taking a specific action a in a given state s. The Q-value function is defined as $Q(s, a)$, and the optimal Q-value function $Q^*(s, a)$ can be found by solving the Bellman equation:

$$Q^*\left(s_t, a_t\right) = \mathbb{E}_{s_{t+1}\sim\mathcal{P}, a_{t+1}\sim\pi}\left[r_t + \gamma \max_{a'} Q^*\left(s_{t+1}, a'\right)\right]$$

where r_t is the reward at time t, γ is the discount factor, $\mathcal{P}$ is the transition function, and π is the policy. The optimal policy can then be derived from Q^* by choosing the action with the highest Q-value at each state.

In DQL, a deep neural network is used to estimate the Q-value function by taking the state s_t and action a_t as inputs:

$$Q\left(s_t, a_t; \theta\right) \approx Q^*\left(s_t, a_t\right)$$

where θ denotes the weights of the neural network. The neural network is trained using stochastic gradient descent to minimize the following loss function:

$$L(\theta) = \mathbb{E}_{s_t, a_t, r_t, s_{t+1}\sim\mathcal{P}}\left[\left(r_t + \gamma \max_{a'} Q\left(s_{t+1}, a'; \theta^-\right) - Q\left(s_t, a_t; \theta\right)\right)^2\right],$$

where θ^{-} denotes a separate set of weights used to calculate the target Q-value. The target network is periodically updated to match the main network weights. The experience replay buffer is also used to store a set of recent experiences, which is then randomly sampled to generate batches for training the neural network.

DQL has been successfully applied in various domains, such as playing Atari games and controlling robotic systems, and has achieved state-of-the-art results in many benchmarks.

2.2.2 Policy-based Methods

Policy-based methods are another category of reinforcement learning algorithms. Unlike value-based methods, which learn the value function and then derive a policy from it, policy-based methods directly learn a policy.

Policy-based methods can be divided into two types: deterministic policy methods and stochastic policy methods. Deterministic policy directly outputs a deterministic action given a state, while stochastic policy outputs a distribution over actions given a state.

The goal of policy-based methods is to find a policy that maximizes the expected return:

$$\max_{\theta} J(\theta) = \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{\infty} \gamma^t r\left(s_t, a_t\right)\right],$$

where π_θ is the policy parameterized by θ. The optimization of policy-based methods is based on the policy gradient theorem.

Theorem 2.3 (Policy gradient theorem). *The gradient of $\mathcal{J}$ is*

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{\infty} \nabla_\theta \log \pi_\theta\left(a_t \mid s_t\right) Q^{\pi_\theta}\left(s_t, a_t\right)\right].$$

Proof.

$$\begin{aligned}
\nabla_\theta J(\pi_\theta) &= \nabla_\theta \int_{\mathcal{S}} d^{\pi_\theta}(s) \int_{\mathcal{A}} \pi_\theta(a|s) Q^{\pi_\theta}(s,a) da ds \\
&= \int_{\mathcal{S}} d^{\pi_\theta}(s) \int_{\mathcal{A}} \nabla_\theta \pi_\theta(a|s) Q^{\pi_\theta}(s,a) da ds && \text{(swap derivative and integral)} \\
&= \int_{\mathcal{S}} d^{\pi_\theta}(s) \int_{\mathcal{A}} \pi_\theta(a|s) \frac{\nabla_\theta \pi_\theta(a|s)}{\pi_\theta(a|s)} Q^{\pi_\theta}(s,a) da ds && \text{(multiply by 1)} \\
&= \mathbb{E}_{\pi_\theta}\left[Q^{\pi_\theta}(s,a) \frac{\nabla_\theta \pi_\theta(a|s)}{\pi_\theta(a|s)} \right] \\
&= \mathbb{E}_{\pi_\theta}\left[Q^{\pi_\theta}(s,a) \nabla_\theta \log \pi_\theta(a|s) \right] && \text{(definition of logarithm)} \\
&\approx \frac{1}{N} \sum_{i=1}^{N} Q^{\pi_\theta}(s_i,a_i) \nabla_\theta \log \pi_\theta(a_i|s_i) && \text{(sample estimate),}
\end{aligned} \tag{2.10}$$

where s_i and a_i are samples collected from the policy π_θ and N is the number of samples used in the estimate. □

Therefore, the policy gradient update rule is given by:

$$\theta_{k+1} = \theta_k + \alpha \nabla_\theta J\left(\pi_\theta\right) \approx \theta_k + \alpha \frac{1}{N} \sum_{i=1}^{N} Q^{\pi_\theta}\left(s_i, a_i\right) \nabla_\theta \log \pi_\theta\left(a_i \mid s_i\right),$$

where α is the learning rate.

REINFORCE [10]

REINFORCE is a policy-based method for reinforcement learning that is based on the policy gradient theorem. In REINFORCE, the goal is to optimize a parametrized policy π_θ with respect to the expected return $J(\theta)$.

The update rule in REINFORCE is as follows:

$$\theta_{t+1} = \theta_t + \alpha \nabla_\theta J\left(\theta_t\right)$$

where α is the step size and $\nabla_\theta J(\theta_t)$ is the gradient of the expected return with respect to the policy parameters.

To derive the gradient of the expected return, we can use the policy gradient theorem:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau \sim p_\theta(\tau)}\left[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta\left(a_t \mid s_t\right) R(\tau)\right]$$

where $\tau = (s_0, a_0, r_1, s_1, a_1, r_2, ..., s_T, a_T, r_{T+1})$ is a trajectory, $p_\theta(\tau)$ is the probability density of trajectory τ under policy π_θ, and $R(\tau)$ is the return of trajectory τ.

Using the policy gradient theorem, we can update the policy parameters as follows:

$$\theta_{t+1} = \theta_t + \alpha \sum_{t=0}^{T} \nabla_\theta \log \pi_\theta \left(a_t \mid s_t\right) R(\tau)$$

where $\tau = (s_0, a_0, r_1, s_1, a_1, r_2, ..., s_T, a_T, r_{T+1})$ is a trajectory sampled from the current policy π_{θ_t}.

One way to implement REINFORCE is to use Monte Carlo simulation to estimate the expected return and the gradient. At each iteration, the agent interacts with the environment to collect a set of trajectories using the current policy, and then uses these trajectories to estimate the gradient of the expected return with respect to the policy parameters. This gradient is then used to update the policy parameters using the update rule above.

In practice, REINFORCE has high variance and is not very sample efficient. To address this issue, several techniques have been proposed, such as using a baseline function to reduce variance and using actor-critic methods to estimate the value function and the policy gradient jointly.

Trust Region Policy Optimization (TRPO) [11]

Trust Region Policy Optimization is a policy optimization method that seeks to improve policy performance while ensuring the policy changes are small enough to guarantee a certain amount of improvement. The idea behind TRPO is to construct a surrogate objective function that approximates the true objective function while ensuring that the new policy is not too far away from the old policy. The optimization is carried out by solving a constrained optimization problem, where the constraints are given by a trust region around the current policy.

The objective function of TRPO is with an additional constraint on the KL-divergence between the new and old policies:

$$\begin{gathered}\max_\theta \mathbb{E}_{s_t, a_t \sim \pi_{\theta_{dd}}}\left[\frac{\pi_\theta\left(a_t \mid s_t\right)}{\pi_{\theta_{dd}}\left(a_t \mid s_t\right)} A^{\pi_{\theta_{dd}}}\left(s_t, a_t\right)\right] \\ \text{subject to } \mathrm{KL}\left[\pi_{\theta_{ddd}}(\cdot \mid s) \| \pi_\theta(\cdot \mid s)\right] \leq \delta\end{gathered}$$

where $A^{\pi_{\theta_{old}}}(s_t, a_t)$ is the advantage function estimated using the old policy, and δ is a hyperparameter that controls the size of the trust region.

To optimize the objective function while satisfying the trust region constraint, TRPO uses a second-order optimization method, such as the conjugate gradient (CG) method, to solve the following constrained optimization problem:

$$\begin{aligned}
&\max_{\theta} \mathbb{E}_{s_t, a_t \sim \pi_{\theta_{old}}}\left[\frac{\pi_\theta\left(a_t \mid s_t\right)}{\pi_{\theta_{old}}\left(a_t \mid s_t\right)} A^{\pi_{\theta_{old}}}\left(s_t, a_t\right)\right] \\
&\text{subject to } \mathbb{E}_{s_t, a_t \sim \pi_{\theta_{old}}}\left[\mathrm{KL}\left[\pi_{\theta_{dd}}\left(\cdot \mid s_t\right) \| \pi_\theta\left(\cdot \mid s_t\right)\right]\right] \leq \delta \\
&\mathbb{E}_{s_t, a_t \sim \pi_{\theta_{old}}}\left[\left(\frac{\pi_\theta\left(a_t \mid s_t\right)}{\pi_{\theta_{old}}\left(a_t \mid s_t\right)}\right)^2\right] \leq \beta
\end{aligned}$$

where β is another hyperparameter that limits the magnitude of policy updates.

Proximal Policy Optimization (PPO) [12]

PPO (Proximal Policy Optimization) is a policy-based method for reinforcement learning. It is a variant of TRPO (Trust Region Policy Optimization) and uses a clipped surrogate objective function to update the policy in each iteration. The objective function in PPO is as follows:

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t\left[\min\left(r_t(\theta)\hat{A}_t, \operatorname{clip}\left(r_t(\theta), 1-\epsilon, 1+\epsilon\right)\hat{A}_t\right)\right],$$

where θ is the parameter of the policy, $\hat{\mathbb{E}}t$ denotes the empirical expectation taken over a batch of data, $\hat{A}t$ is the advantage estimate at time step t, and $r_t(\theta) = \frac{\pi\theta(a_t|s_t)}{\pi\theta_{old}(a_t|s_t)}$ is the ratio of the probability of taking action a_t in state s_t under the new and old policies.

The clipped surrogate objective function consists of two terms: the first term is the advantage-weighted surrogate objective, which encourages the new policy to increase the probability of taking actions that have a high advantage; the second term is a clipped surrogate objective, which prevents the new policy from deviating too much from the old policy.

The advantage estimate $\hat{A}_t$ is used to estimate the advantage function $A(s_t, a_t)$. In PPO, the generalized advantage estimator (GAE) is used to estimate $\hat{A}_t$:

$$\hat{A}_t = \delta_t + \gamma\lambda\delta_{t+1} + \ldots + \gamma^{T-t+1}\lambda^{T-t}\delta_T,$$

where $\delta_t = r_t + \gamma V(s_{t+1}) - V(s_t)$ is the TD error at time step t, γ is the discount factor, $V(s)$ is the state value function, and λ is a hyperparameter that controls the trade-off between bias and variance in the estimation of the advantage function.

The policy is updated by minimizing the clipped surrogate objective function subject to a constraint on the maximum change in the policy parameters:

$$\hat{A}_t = \delta_t + \gamma\lambda\delta_{t+1} + \ldots + \gamma^{T-t+1}\lambda^{T-t}\delta_T,$$

where θ' are the updated policy parameters and ϵ is a hyperparameter that controls the size of the trust region.

The update rule for the policy parameters is:

$$\theta_{k+1} = \arg\max_{\theta} \frac{1}{N}\sum_{i=1}^{N} \min\left(\frac{\pi_\theta\left(a_i \mid s_i\right)}{\pi_{\theta_{\text{odd}}}\left(a_i \mid s_i\right)}\hat{A}_i, \operatorname{clip}\left(\frac{\pi_\theta\left(a_i \mid s_i\right)}{\pi_{\theta_{\text{old}}}\left(a_i \mid s_i\right)}, 1-\epsilon, 1+\epsilon\right)\hat{A}_i\right).$$

2.2.3 Actor-Critic [1]

The actor-critic method is a combination of policy-based and value-based methods. It uses a policy (actor) network to select actions and a value (critic) network to evaluate the selected actions. The actor-critic method addresses some of the limitations of both policy-based and value-based methods. In particular, it can handle continuous action spaces and has improved sample efficiency compared to policy-based methods.

In the actor-critic method, the policy network is typically updated using the policy gradient method, and the value network is updated using the TD learning method. The policy gradient method uses the gradient of the policy with respect to the expected return to update the policy network parameters. The TD learning method uses the difference between the observed return and the predicted return to update the value network parameters.

The critic learns the value function $V_\theta(s)$ which estimates the expected return starting from state s:

$$V_\theta(s) \approx \mathbb{E}\left[G_t \mid s_t = s\right]$$

, where $G_t = \sum_{k=t+1}^{T} \gamma^{k-t-1} r_k$ is the discounted sum of rewards starting from time step t, and T is the time step at which the episode terminates.

$$\pi_\phi(a \mid s) \approx \mathbb{P}\left[a_t = a \mid s_t = s\right]$$

, where $G_t = \sum_{k=t+1}^{T} \gamma^{k-t-1} r_k$ is the discounted sum of rewards starting from time step t, and T is the time step at which the episode terminates.

The actor updates its policy parameters ϕ in the direction of the policy gradient:

$$\nabla_\phi J(\phi) \approx \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \nabla_\phi \log \pi_\phi \left(a_t^i \mid s_t^i\right) Q^\pi \left(s_t^i, a_t^i\right),$$

where $Q^\pi(s_t^i, a_t^i)$ is an estimate of the expected sum of rewards starting from state s_t^i, taking action a_t^i and then following policy π thereafter. This estimate is provided by the critic, and is updated using TD learning as follows:

$$\begin{aligned} &\delta_t = r_t + \gamma V_\theta\left(s_{t+1}\right) - V_\theta\left(s_t\right) \\ &\theta \leftarrow \theta + \alpha_\theta \delta_t \nabla_\theta V_\theta\left(s_t\right), \end{aligned}$$

where δ_t is the TD error at time step t, α_θ is the learning rate for the critic, and $\nabla_\theta V_\theta(s_t)$ is the gradient of the value function with respect to its parameters θ.

The actor's policy parameters ϕ are updated in the direction of the policy gradient, scaled by the advantage function:

$$\phi \leftarrow \phi + \alpha_\phi A_t \nabla_\phi \log \pi_\phi\left(a_t \mid s_t\right),$$

where A_t is the advantage function, defined as $A_t = Q^\pi(s_t, a_t) - V_\theta(s_t)$. The advantage function measures the advantage of taking action a_t in state s_t compared to following the current policy.

DDPG [13]

DDPG stands for Deep Deterministic Policy Gradient, and it is a model-free, off-policy algorithm that combines ideas from both Q-learning and policy gradient methods. DDPG is designed for continuous action spaces, which makes it a popular choice for applications such as robotics and control.

DDPG uses an actor-critic architecture, where the actor network learns a deterministic policy, and the critic network learns the corresponding action value function. The critic network is trained using the Bellman equation, similar to Q-learning, and the actor network is updated using the gradient of the critic network with respect to the actor's output, similar to policy gradient methods.

The update rule for the critic network is similar to Q-learning, and it is defined as follows:

$$Q\left(s_t, a_t\right) \leftarrow Q\left(s_t, a_t\right)+\alpha\left(r_t+\gamma Q\left(s_{t+1}, \mu\left(s_{t+1}\right)\right)-Q\left(s_t, a_t\right)\right)$$

where s_t is the current state, a_t is the current action, r_t is the immediate reward, $\mu\left(s_t\right)$ is the actor's output for the current state, α is the learning rate, and γ is the discount factor.

The actor network is updated using the gradient of the critic network with respect to the actor's output. The update rule for the actor network is as follows:

$$\Delta \theta \leftarrow \frac{1}{N} \sum_{i=1}^N \left.\nabla_a Q\left(s, a \mid \theta^Q\right)\right|_{s=s_i, a=\mu\left(s_i\right)} \left.\nabla_{\theta^\mu} \mu\left(s \mid \theta^\mu\right)\right|_{s=s_i}$$

where θ^Q and θ^μ are the weights of the critic and actor networks, respectively, and N is the batch size.

DDPG also uses a replay buffer and a target network to improve the stability of the learning process. The replay buffer stores the experiences of the agent in the form of tuples $\left(s_t, a_t, r_t, s_{t+1}\right)$, and samples are randomly drawn from the buffer during training. The target network is a copy of the actor and critic networks that is periodically updated with the weights of the online networks to improve the stability of the learning process.

2.3 Multi-agent Reinforcement Learning

Dec-POMDP. A cooperative MARL problem can be formulated as a *decentralized partially observable Markov decision process* (Dec-POMDP) [14], which is described as a tuple $\langle n, \boldsymbol{S}, \boldsymbol{A}, P, R, \boldsymbol{O}, \boldsymbol{\Omega}, \gamma\rangle$, where n represents the number of agents. $\boldsymbol{S}$

represents the space of global states. $\boldsymbol{A} = \{A_i\}_{i=1,\cdots,n}$ denotes the space of actions of all agents. $\boldsymbol{O} = \{O_i\}_{i=1,\cdots,n}$ denotes the space of observations of all agents. $P : \boldsymbol{S} \times \boldsymbol{A} \rightarrow \boldsymbol{S}$ denotes the state transition probability function. All agents share the same reward as a function of the states and actions of the agents $R : \boldsymbol{S} \times \boldsymbol{A} \rightarrow \mathbb{R}$. Each agent i receives a private observation $o_i \in O_i$ according to the observation function $\boldsymbol{\Omega}(s, i) : \boldsymbol{S} \rightarrow O_i$. $\gamma \in [0, 1]$ denotes the discount factor.

Part I

Communication amongst agents

Chapter 3

Learning Efficient Multi-agent Communication: An Information Bottleneck Approach[1]

3.1 Introduction

Communication in Multi-Agent Reinforcement Learning (MARL) refers to the exchange of information or messages among agents during the learning process. It can be explicit or implicit, depending on the underlying communication channel used to exchange information.

In MARL, communication is essential as it enables agents to coordinate their actions and learn better policies. In cooperative environments, agents must work together to achieve a common goal, which can be difficult without communication. Communication allows agents to share information about the environment, their policies, and their observations, which can help them make better decisions and act more efficiently. Therefore, many recent works in the field of multi-agent communication focus on learning what messages [16–18] to send and whom to address them [19–22].

There are different types of communication strategies in MARL, including centralized, decentralized, and hybrid. In centralized communication, agents can directly

[1]The work in this chapter has been published as Wang et al.

exchange information with one another or with a central coordinator. In decentralized communication, agents communicate only with their neighbors or with those in their local communication range. Hybrid communication combines elements of both centralized and decentralized communication.

Communication can help agents coordinate their actions in several ways [23]. First, it can allow agents to learn about the state of the environment and the actions of other agents. This information can be used to coordinate actions, avoid conflicts, and allocate resources more efficiently. Second, communication can enable agents to share knowledge about their individual policies and observations, allowing them to learn from each other and improve their policies. Finally, communication can help agents to negotiate and coordinate their strategies to achieve common goals, leading to better performance and higher rewards.

Communication helps cooperation in MARL in several ways:

- Sharing Information: Communication allows agents to share information with each other, which can improve their collective decision-making. For example, one agent may have observed a relevant event that another agent has not, and by sharing this information, they can improve their understanding of the environment.

- Coordination: Communication can help agents coordinate their actions and avoid conflicts. For example, if two agents are approaching the same goal from different directions, they can communicate to coordinate their movements and avoid colliding with each other.

- Division of Labor: Communication can help agents divide tasks among themselves based on their strengths and weaknesses. For example, one agent may be better suited to a particular task than another, and by communicating, they can divide the work in a way that maximizes their collective performance.

- Joint Planning: Communication can help agents jointly plan their actions to achieve a common goal. For example, if a group of agents is trying to solve a complex problem, they can communicate to develop a joint plan of action that takes into account each agent's individual capabilities.

However, communication in MARL can also be challenging, especially in large-scale environments with many agents. Communication can introduce additional complexity and may require more computational resources. Additionally, communication may be subject to delays, noise, and errors, which can affect the performance of the agents. Therefore, designing effective communication protocols and strategies is an important area of research in MARL.

In this chapter, we address a key challenge faced by learning agents in multi-agent domains, namely the need to efficiently utilize limited communication resources such as limited bandwidth. Limited bandwidth arises in two transmission processes, namely from agents to the scheduler and from the scheduler to agents, as illustrated in Fig. 3.1. Recently, one strategy has been proposed for dealing with the limited bandwidth problem in multi-agent settings, which involves downsizing the communication group via a scheduler [24–26]. This approach allows only a subset of agents to communicate at a given time, thereby avoiding the issue of bandwidth overload. However, this approach suffers from several limitations. For instance, it restricts the number of agents who can communicate, and agents may share redundant messages, which can lead to unsustainable bandwidth usage. Moreover, these methods require specific configurations, such as a predefined scale of agents' communication group [24, 25] or a predefined threshold for muting agents [26], which may not be suitable for complex multi-agent domains.

We propose an approach to address the limited bandwidth problem by compressing the communication messages. First, we view the messages as random vectors from a communication theory perspective, and prove that limited bandwidth can be translated into a constraint on the communicated message entropy. This implies that agents should generate low-entropy messages to satisfy the limited bandwidth constraint. Specifically, we apply the source coding theorem [27] and Nyquist criterion [28] to state that, in a noiseless channel, when a K-ary, bandwidth B, quantization interval Δ communication system transmits n messages of dimension d per second, the entropy of the messages $H(\boldsymbol{m})$ is limited by the bandwidth according to $H(\boldsymbol{m}) \leq \frac{2B\log_2 K}{n} + d\log_2 \Delta$.

To enable agents to send and receive low-entropy messages that contain useful and necessary information, we propose a method for learning communication protocols and scheduling, called *Informative Multi-Agent Communication* (IMAC). IMAC is inspired by the variational information bottleneck method [29, 30], and it applies a regularization method to learn informative communication protocols that convey low-entropy and useful messages. Specifically, IMAC applies the variational information bottleneck to the communication protocol by viewing the messages as latent variables and approximating their posterior distribution. By regularizing the mutual information between the protocol's inputs (i.e., the input features extracted from agents) and the protocol's outputs (i.e., the messages), we learn informative communication protocols. Additionally, we view the scheduler as a virtual agent and learn a weight-based scheduler that aggregates compact messages by reweighting all agents' messages.

We conduct extensive experiments in different environments: cooperative navigation, predator-prey and StarCraftII. Results show that IMAC can convey low-entropy messages, enable effective communication among agents under the limited bandwidth constraint, and lead to faster convergence as compared with various baselines.

3.2 Related Work

Our work is related to previous studies in multi-agent reinforcement learning with communication, which primarily focus on two fundamental issues: who/whom and what to communicate, also referred to as learning scheduling and communication protocols. One line of scheduling methods uses specialized networks to learn a weight-based scheduler by reweighting agents' messages, such as bi-direction RNNs in BiCNet [18] or a self-attention layer in TarMAC [21]. Another line introduces various gating mechanisms to determine the groups of communication agents [19, 20, 22, 25, 26]. Communication protocols are usually learned in an end-to-end manner with a specific scheduler, going from perceptual input (e.g., pixels) to communication symbols (discrete or continuous) to actions (e.g., navigating in an environment) [16, 25]. While some studies on learning communication protocols concentrate on discrete, human-interpretable communication symbols [31, 32], our approach learns a continuous communication protocol implicitly [16, 17, 19, 22].

Various methods have been proposed to address the limited bandwidth problem, such as downsizing the communication group via a scheduler. However, all scheduling methods are subject to content redundancy, which is unsustainable under

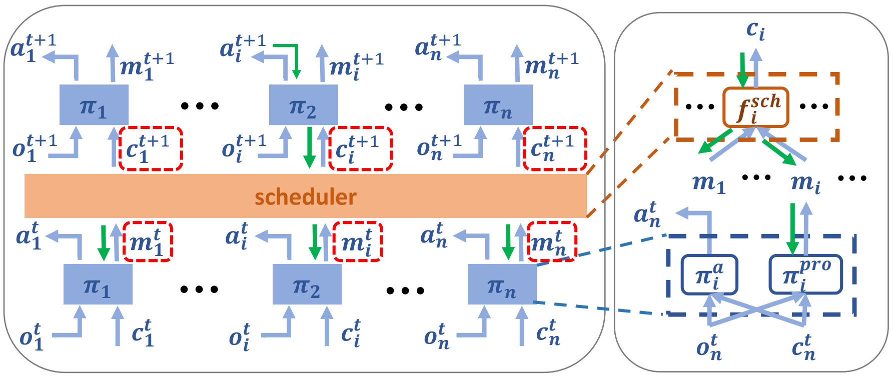


FIGURE 3.1: The Architecture of IMAC. **Left:** Overview of the communication scheme. The red dashed box means the communication process with a limited bandwidth constraint. The green line means the gradient flows. **Right:** The upper one is the scheduler for agent i. The below one is the policy π_i^a and the communication protocol network π_i^{pro} for agent i

.

bandwidth limitations. Even if only a single pair of agents can communicate, a large message may not be conveyed due to the limited bandwidth. Furthermore, scheduling methods with gating mechanisms are inflexible as they introduce manual configurations, such as the predefined size of a communication group [24, 25] or a handcrafted threshold for muting agents [19, 26]. Additionally, most methods for learning communication protocols fail to compress the protocols and extract valuable information for cooperation [19]. In this paper, we investigate the limited bandwidth from the communication protocols' perspective. Furthermore, our methods can be extended to scheduling by utilizing a weight-based scheduler.

In recent years, the combination of the information bottleneck method and reinforcement learning has led to several applications, particularly in imitation learning [33], inverse reinforcement learning [33], and exploration [3, 34]. Among them, Goyal et al. mentioned multi-agent communication in their appendix, demonstrating a method to minimize communication by penalizing the effect of one agent's messages on another agent's policy. However, it does not consider the limited bandwidth constraint.

3.3 Communicative Multi-agent Reinforcement Learning

We consider a communicative multi-agent reinforcement learning task [35], which is extended from Dec-POMDP and described as a tuple $\langle n, \boldsymbol{S}, \boldsymbol{A}, r, P, \boldsymbol{O}, \boldsymbol{\Omega}, \boldsymbol{M}, \gamma \rangle$, where n represents the number of agents. $\boldsymbol{S}$ represents the space of global states. $\boldsymbol{A} = \{A_i\}_{i=1,\cdots,n}$ denotes the space of actions of all agents. $\boldsymbol{O} = \{O_i\}_{i=1,\cdots,n}$ denotes the space of observations of all agents. $\boldsymbol{M}$ represents the space of messages. $P : \boldsymbol{S} \times \boldsymbol{A} \to \boldsymbol{S}$ denotes the state transition probability function. All agents share the same reward as a function of the states and agents' actions $r : \boldsymbol{S} \times \boldsymbol{A} \to \mathbb{R}$. Each agent i receives a private observation $o_i \in O_i$ according to the observation function $\boldsymbol{\Omega}(s, i) : \boldsymbol{S} \to O_i$. $\gamma \in [0, 1]$ denotes the discount factor. As shown in Fig. 3.1, each agent receives observation o_i and a scheduling message c_i, then outputs an action a_i and a message m_i. A scheduler f_i^{sch} is introduced to receive messages $[m_1, \cdots, m_n] \in \boldsymbol{M}$ from all agents and dispatch scheduling messages $c_i = f_i^{sch}(m_1, \cdots, m_n) \in M_i$ for each agent i.

We adopt a centralized training and decentralized execution paradigm [36], and further relax it by allowing agents to communicate. That is, during training, agents are granted access to the states and actions of other agents for the centralized critic, while decentralized execution only requires individual states and scheduled messages from a well-trained scheduler.

Our end-to-end method is to learn a communication protocol $\pi_i^{pro}(m_i|o_i, c_i)$, an policy $\pi_i^a(a_i|o_i, c_i)$, and a scheduler $f_i^{sch}(c_i|m_1, \cdots, m_n)$, which jointly maximize the expected discounted return for each agent i:

$$\begin{aligned} J_i &= \mathbb{E}_{\pi_i^a, \pi_i^{pro}, f_i^{sch}}[\Sigma_{t=0}^{\infty} \gamma^t r_i^t(s, a)] \\ &= \mathbb{E}_{\pi_i^a, \pi_i^{pro}, f_i^{sch}}[Q_i(s, a_1, \cdots, a_n)] \\ &\approx \mathbb{E}_{\pi_i^a, \pi_i^{pro}, f_i^{sch}}[Q_i(o_1, \cdots, o_n, c_1, \cdots, c_n, a_1, \cdots, a_n)] \\ &= \mathbb{E}_{\pi_i^a, \pi_i^{pro}, f_i^{sch}}[Q_i(h_1, \cdots, h_n, a_1, \cdots, a_n)] \end{aligned} \tag{3.1}$$

where r_i^t is the reward received by the $i-$th agent at time t, Q_i is the centralized action-value function for each agent i, and $\mathbb{E}_{\pi_i^a, \pi_i^{pro}, f_i^{sch}}$ denotes an expectation over the trajectories $\langle s, a_i, m_i, c_i \rangle$ generated by

$p^{\pi_i^a}, \pi_i^a(a_i|o_i, c_i), \pi_i^{pro}(m_i|o_i, c_i), f_i^{sch}(c_i|m_1, \cdots, m_n)$. Here we follow the simplification in [36] to replace the global states with joint observations and use an abbreviation h_i to represent the joint value of $[o_i, c_i]$ in the rest of this chapter.

The limited bandwidth B is a range of frequencies within a given band. It exists in the two processes of transmission: messages from agents to the scheduler and scheduling messages from the scheduler to agents as shown in Fig. 3.1. In the next section, we will discuss how the limited bandwidth B affects the communication.

3.4 Connection between Limited Bandwidth and Multi-agent Communication

In this section, from the perspective of communication theory, we show how the limited bandwidth B requires low-entropy messages throughout the transmission. We then discuss how to measure the message's entropy.

3.4.1 Communication Process

Figure 3.2 illustrates the communication process from agents to the scheduler, which involves five stages: analog-to-digital conversion, coding, transmission, decoding, and digital-to-analog conversion [28]. When an agent transmits a continuous message m_i, it is first mapped into a countable set through an analog-to-digital converter (ADC). The ADC can be modeled as two processes: sampling, which converts a time-varying signal into a sequence of discrete-time real-value signals corresponding to the discrete timestep in RL, and quantization, which replaces each real number with an approximation from a finite set of discrete values. In the coding phase, the quantized messages m_i^{Δ} are mapped to a bitstream using source coding methods such as Huffman coding. During the transmission phase, the transmitter modulates the bitstream into a wave, which is then transmitted through a channel. The receiver demodulates the wave into another bitstream, but there may be some distortion in the channel. The decoding phase is the inverse operation of coding, mapping the bitstream to the recovered messages $\hat{m}_i^{\Delta}$. Finally, the scheduler receives a reconstructed analog message from a digital-to-analog converter (DAC). The bandwidth restricts the transmission phase, which is the most critical stage for communication in terms of limited resources.

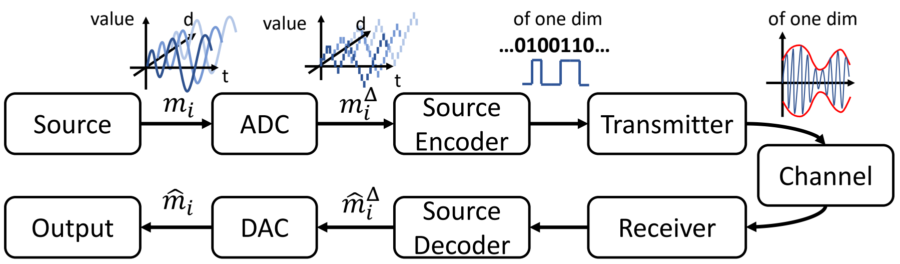


Figure 3.2: Overview of the Communication Process. Axes of messages are time, dimension of the message vector, and value of each element in the vector.

3.4.2 Limited Bandwidth Restricts Message's Entropy

We model the messages m_i as continuous random vectors M_i, i.e., continuous vectors sampled from a certain distribution. The reason is that a message is sent by one agent in each timestep, while over a long duration, the messages follow some distributions. We abuse $\boldsymbol{m}$ to represent the random vector by omitting the subscript, and explain the subscript in special cases.

For simplicity, we consider sending an element X of the continuous random vector $\boldsymbol{m}$, which is a continuous random variable, and then extend our conclusion to $\boldsymbol{m}$. First, we quantize the continuous variable into discrete symbols. The quantization brings a gap between the entropy of the discrete variables and differential entropy of the continuous variables.

Remark 3.1 (Relationship between entropy and differential entropy). Consider a continuous random variable X with a probability density function $f_X(x)$. This function is then quantized by dividing its range into K levels of interval Δ, where $K = ceil(|X|/\Delta)$, and $|X|$ is max amplitude of variable. The quantized variable is X^Δ. Then the difference between differential entropy $H(X)$ and entropy $H\left(X^\Delta\right)$ is $H(X) - H\left(X^\Delta\right) = \log_2(\Delta)$.

Note that for a fixed small identical interval Δ, there is only a constant difference between differential entropy and entropy. Then we encode these quantized symbols.

Remark 3.2 (Source Coding Theorem [27]). In the source coding phase, a set of n quantized symbols is to be encoded into bitstreams. These symbols can be treated as n independent samples of a discrete random variable X^{Δ} with entropy $H(X^{\Delta})$. Let L be the average number of bits to encode the n symbols. The minimum L satisfies $H(X^{\Delta}) \leq L < H(X^{\Delta}) + \frac{1}{n}$.

Remark 3.2 regularizes the coding phase in the communication process. Then in the transmission, over a noiseless channel, the maximum bandwidth is defined as the maximum data rate:

Remark 3.3 (The Maximum Data Rate [28]). The maximum data rate R_{max} (bits per second) over a noiseless channel satisfies: $R_{max} = 2B\log_2 K$, where B is the bandwidth (Hz) and K is the number of signal levels.

Remark 3.3 is derived from the Nyquist criterion [28] and specifies how the bandwidth of a communication system affects the transmission data rate for reliable transmission in the noiseless condition. Based on these three remarks, we show how the limited bandwidth constraint affects the multi-agent communication.

Proposition 3.1. *In a noiseless channel, the bandwidth of channel B limits the entropy of the messages' elements.*

Proof. Given a message's element X as an i.i.d continuous random variable with differential entropy $H(X)$, its quantized time series $X_1^{\Delta}, \cdots, X_t^{\Delta}, \cdots$ (here the subscript means different times) with entropy $H(X^{\Delta}) = H(X) - \log_2 \Delta$, the communication system's bandwidth B, as well as the signal levels K, the communication system transmits n symbols per second. We define R_{code} as an unbiased estimation of L in Remark 2. So the transmission rate $R_{trans}(\frac{\text{bit}}{\text{second}}) = n \cdot R_{code}(\frac{\text{bit}}{\text{symbol}}) \geq n \cdot H(X^{\Delta}) \geq n \cdot (H(X) - \log \Delta)$.[2] According to Remark 3.3, $R_{trans} \leq R_{max} = 2B\log_2 K$. Consequently, we have $H(X) \leq \frac{2B\log_2 K}{n} + \log_2 \Delta$. $\square$

Note that although a frequent symbol among these sending symbols uses less bits than $H(X^{\Delta})$ and vice versa, when we send a bunch of symbols, R_{code} is lager than $H(X^{\Delta})$ *on average.*

Proposition 3.2. *In a noiseless channel, the bandwidth of channel B limits the entropy of the messages.*

[2] $\frac{\text{bit}}{\text{second}}$ and $\frac{\text{bit}}{\text{symbol}}$ are units of measure.

Proof. When sending the random vector, i.e., the message $M_i = [X_1, X_2, \cdots, X_d]$, where the subscript means different entries of the vector and d is the dimension, each variable X_i occupies a bandwidth B_i, which satisfies $\sum_{i=1}^{d} B_i = B$. Assume all entries are quantilized with the same interval, according to [37], the upper bound of the messages $H(M_i) = H(X_1, \cdots, X_d) \leq \sum_{i=1}^{d} H(X_i) \leq \frac{2dB\log_2 K}{n} + d\log_2 \Delta$. □

Eventually, a limited bandwidth B enforces an upper bound H_c to the message's entropy $H(M_i) \leq H_c \propto B$.

3.4.3 Measurement of a Message's Entropy

The messages M_i as an i.i.d random vector can follow any distribution, so it is hard to determine the message's entropy. So, we keep a historical record of the message and find a quantity to measure the message's entropy.

Proposition 3.3. *When we have a historical record of the messages to estimate the messages' mean $\boldsymbol{\mu}$ and co-variance $\boldsymbol{\Sigma}$, the entropy of the messages is upper bounded by $\frac{1}{2}\log((2\pi e)^d|\boldsymbol{\Sigma}|)$, where d is the dimension of M_i.*

Proof. The message M_i follows a certain distribution, and we are only certain about its mean and variance. According to the principle of maximum entropy [38], the Gaussian distribution has maximum entropy relative to all probability distributions covering the entire real line $(-\infty, \infty)$ but having a finite mean and finite variance (see the proof in [37]). So $H(M_i) \leq \frac{1}{2}\log((2\pi e)^d\boldsymbol{\Sigma})$, where the right term is the entropy of multivariate Gaussian $N(\boldsymbol{\mu}, \boldsymbol{\Sigma})$. □

We conclude that $\frac{1}{2}\log((2\pi e)^d|\boldsymbol{\Sigma}|)$ offers an upper bound to approximate $H(M_i)$, and this upper bound should be less than or equal to the limited bandwidth constraint to ensure that the message with any possible distribution satisfies the limited bandwidth constraint.

3.5 Informative Multi-agent Communication

As shown in the previous section, the limited bandwidth requires agents to send low-entropy messages. In this section, we first introduce our method for learning a valuable and low-entropy communication protocol via the information bottleneck principle. Then, we discuss how to use the same principle in scheduling.

3.5.1 Variational Information Bottleneck for Learning Protocols

We propose an informative multi-agent communication via information bottleneck principle to learn protocols. Concretely, we propose an information-theoretic regularization on the mutual information $I(H_i; M_i)$ between the messages and the input features

$$I(H_i; M_i) \leq I_c \tag{3.2}$$

where M_i is a random vector with a probability density function $p_{M_i}(m_i)$, which represents the possible assignments of the messages m_i, and H_i is a random vector with a probability density function $p_{H_i}(h_i)$ which the possible values of $[o_i, c_i]$. We omit the subscripts in the density functions in the rest of the paper. Eventually, with the help of variational information bottleneck [30], this regularization enforces agents to send low-entropy messages.

Consider a scenario with n agents' policies $\{\pi_i^a\}_{i=1,\cdots,n}$ and protocols $\{\pi_i^{pro}\}_{i=1,\cdots,n}$ which are parameterized by $\{\theta_i\}_{i=1,\cdots,n} = \{\theta_i^a, \theta_i^{pro}\}_{i=1,\cdots,n}$, and with schedulers $\{f_i^{sch}\}_{i=1,\cdots,n}$ which are parameterized by $\{\phi_i\}_{i=1,\cdots,n}$. Consequently, for learning the communication protocol with fixed schedulers, the agent i is supposed to maximize:

$$\begin{gathered} J(\theta_i) = \mathbb{E}_{\pi_i^a, \pi_i^{pro}, f_i^{sch}}[Q_i(h_1, \cdots, h_n, a_1, \cdots, a_n)] \\ \text{s.t.} \;\; I(H_i; M_i) \leq I_c. \end{gathered}$$

Practically, we propose to maximize the following objective using the information bottleneck Lagrangian:

$$J'(\theta_i) = \mathbb{E}_{\pi_i^a, \pi_i^{pro}, f_i^{sch}}[Q_i(h_1, \cdots, h_n, a_1, \cdots, a_n)] - \beta I(H_i; M_i), \tag{3.3}$$

where the β is the Lagrange multiplier. The mutual information is defined according to:

$$\begin{aligned} I(H_i; M_i) &= \iint p(h_i, m_i) \log \frac{p(h_i, m_i)}{p(h_i)p(m_i)} \mathrm{d}h_i \mathrm{d}m_i \\ &= \iint p(h_i) \pi^{pro}(m_i|h_i) \log \frac{\pi^{pro}(m_i|h_i)}{p(m_i)} \mathrm{d}h_i \mathrm{d}m_i, \end{aligned}$$

where $p(h_i, m_i)$ is the joint probability of h_i and m_i.

However, computing the marginal distribution $p(m_i) = \int \pi^{pro}(m_i|h_i)p(h_i)dh_i$ can be challenging since we do not know the prior distribution of $p(h_i)$. With the help of variational information bottleneck [30], we use a Gaussian approximation $z(m_i)$ of the marginal distribution $p(m_i)$ and view π^{pro} as multivariate variational encoders. Since $D_{KL}[p(m_i)||z(m_i)] \geq 0$, where the D_{KL} is the Kullback-Leibler divergence, we expand the KL term and get $\int p(m_i) \log p(m_i) dm_i \geq \int p(m_i) \log z(m_i) dm_i$, an upper bound on the mutual information $I(H_i; M_i)$ can be obtained via the KL divergence:

$$\begin{aligned} I(H_i; M_i) &\leq \int p(h_i) \pi_i^{pro}(m_i|h_i) \log \frac{\pi_i^{pro}(m_i|h_i)}{z(m_i)} dh_i dm_i \\ &= \mathbb{E}_{h_i \sim p(h_i)}[D_{KL}[\pi^{pro}(m_i|h_i) \| z(m_i)]]. \end{aligned} \tag{3.4}$$

This provides a lower bound $\tilde{J}(\theta)$ on the regularized objective that we maximize:

$$\begin{aligned} J'(\theta_i) \geq \tilde{J}(\theta_i) = \mathbb{E}_{\pi_i^a, \pi_i^{pro}, f_i^{sch}}&[Q_i(h_1, \cdots, h_n, a_1, \cdots, a_n)] \\ &-\beta \mathbb{E}_{h_i \sim p(h_i)}[D_{KL}[\pi_i^{pro}(m_i|h_i) \| z(m_i)]]. \end{aligned}$$

Consequently, the objective's derivative is:

$$\begin{aligned} \nabla_{\theta_i} \tilde{J}(\pi_i) = \mathbb{E}_{\pi_i^a, \pi_i^{pro}, f_i^{sch}} \Big[&\nabla_{\theta_i} \log(\pi_i(a_t|s_t)) Q_i(h_1, \cdots, h_n, a_1, \cdots, a_n) \\ &- \beta \nabla_{\theta_i} D_{KL}[\pi^{pro}(m_i|h_i) \| z(m_i)] \Big]. \end{aligned} \tag{3.5}$$

With the variational information bottleneck, we can control the messages' distribution and thus control their entropy with different prior $z(m_i)$ to satisfy different limited bandwidths in the training stage. That is, with the regulation of $D_{KL}[p(m_i|h_i)\|z(m_i)]$, the messages' probability density function $p(m_i) =$

$\sum_{h_i} p(m_i|h_i)p(h_i) \approx \sum_{h_i} z(m_i)p(h_i) = z(m_i)\sum_{h_i} p(h_i) = z(m_i)$. Thus $H(M_i) = -\int p \log p \mathrm{d}m_i \approx -\int z \log z \mathrm{d}m_i$.

3.5.2 Unification of Learning Protocols and Scheduling

The scheduler for agent i is $f_i^{sch}(c_i|m_1, \cdots, m_n)$. The terms "scheduler are from SchedNet [25], which introduces "communication scheduling" and "scheduler" to represent the filtering process instead of timing. Recall the communication protocols for agent i: $\pi_i^{pro}(m_i|h_i)$. Due to the same form of the protocol and the scheduling, the scheduler is supposed to follow the same principle for learning a weight-based mechanism. Variational information bottleneck can be applied in scheduling for agent i with regularization on the mutual information between the scheduling messages c_i and all agents' messages $I(C_i; M_1, \cdots, M_n)$, where C_i is a random vector with a probability density function $p_{C_i}(c_i)$, which represent different values of c_i. We follow the joint training scheme for training the communication protocol and scheduling [16], which allows the gradients flow across agents from the recipient agent to the scheduler to the sender agent.

Formally, the schedulers $\{f_i^{sch}\}_{i=1,\cdots,n}$ are parameterized by $\{\phi_i\}_{i=1,\cdots,n}$ as defined in section 3.5.1. We would optimize the lower bound in terms of $\{\phi_i\}_{i=1,\cdots,n}$:

$$\begin{aligned} J'(\phi_i) \geq \tilde{J}(\phi_i) = \mathbb{E}_{\pi_i^a, \pi_i^{pro}, f_i^{sch}}[Q_i(h_1, \cdots, h_n, a_1, \cdots, a_n)] \\ -\beta \mathbb{E}_{p(m_1, m_2, \cdots, m_n)}[D_{KL}[f_i^{sch}(c_i|m_1, \cdots, m_n) \| z(c_i)]]. \end{aligned}$$

Consequently, the objective's derivative is:

$$\begin{aligned} \nabla_{\theta_i} \tilde{J}(\phi_i) = \mathbb{E}_{\pi_i^a, \pi_i^{pro}, f_i^{sch}} \Big[\nabla_{\theta_i} \log(\pi_i(a_t|s_t)) \, Q_i(h_1, \cdots, h_n, a_1, \cdots, a_n) \\ - \beta \nabla_{\phi_i} D_{KL}[f_i^{sch}(c_i|m_1, \cdots, m_n) \| z(c_i)] \Big]. \quad (3.6) \end{aligned}$$

3.5.3 Implementation of the limited bandwidth Constraint

During the execution stage, it is crucial to ensure that the messages transmitted among agents satisfy the low-entropy requirement imposed by the limited bandwidth. We enforce this constraint by clipping the variance of the messages.

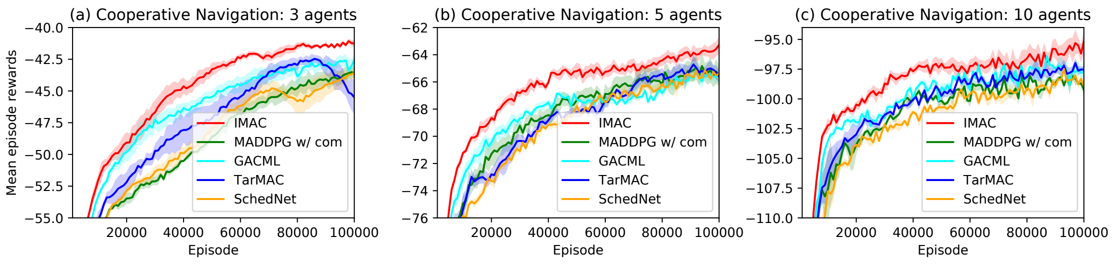


FIGURE 3.3: Learning curves comparing IMAC to other methods for cooperative navigation. As the number of agents increases (from left to right), IMAC improves agents' performance and converge faster.

However, in real-life communication scenarios, the amount of information that a bitstream can carry may vary depending on the source coding method and the communication protocol used. Therefore, we use entropy as a general measure to simulate the limited-bandwidth constraint. To achieve this, we use a batch-normalization-like layer that records the mean and variance of the messages during training, as required by Proposition 3.2. During execution, the normalization layer clips the messages by reducing their variance to meet the limited-bandwidth constraint. Note that this layer is customized and different from the standard batch normalization [39].

For example, suppose we have a 4-ary communication system with a maximum bandwidth of 100 bit/s and want to achieve reliable transmission while transmitting 1000 messages per second. Then, according to $\frac{1}{2}\log(2\pi e\sigma^2) = \frac{2B\log_2 K}{n}$, we can determine the equivalent variance $\sigma^2 \approx 3.2$. During the training stage, we record the variance of the agents' messages, which is found to be 5. During the inference stage, the limited bandwidth requires that the entropy of the messages not exceed 3.2. Thus, we decrease the variance of the messages from 5 to 3.2 using the specific normalization layer.

3.6 Experiment

Experimental Settings. In our study, we evaluate the performance of IMAC on various tasks and environments, including cooperative navigation and predator-prey scenarios in the Multi-Particle Environment [36], as well as the 3m and 8m

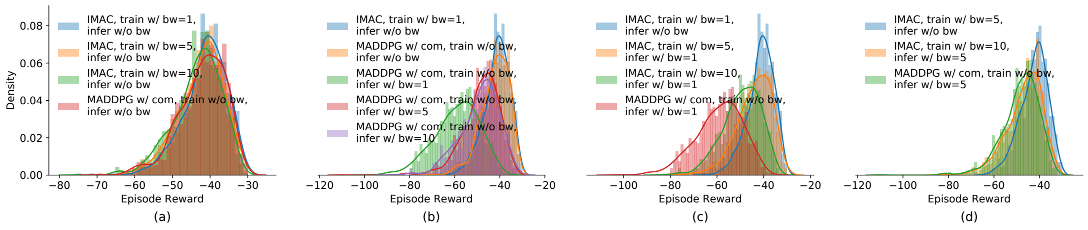


FIGURE 3.4: Density plot of episode reward per agent during the execution stage. (a) Reward distribution of IMAC trained with different prior distributions against MADDPG with communication. (b) Reward distribution of MADDPG with communication under different limited bandwidth environment. (c), (d) Reward distribution of IMAC trained with different prior distributions against MADDPG with communication under the same bandwidth constraint. "bw=δ" means in the implementation of the limited bandwidth constraint, the variance $\mathbf{\Sigma}$ of Gaussian distribution is δ.

scenarios in StarCraft II [40]. Further details regarding the experimental environments are provided in the following subsections, as well as in the supplementary material.

Baseline Methods. To assess the effectiveness of IMAC, we compare it against the following baseline methods: (1) TarMAC [21]: a state-of-the-art multi-agent communication method for limited bandwidth scenarios that employs a self-attention weight-based scheduling mechanism for scheduling and learns the communication protocol in an end-to-end manner, (2) GACML [26]: a multi-agent communication method for limited bandwidth scenarios that uses a gating mechanism for downsizing communication agents and learns the communication protocol in an end-to-end manner, and (3) SchedNet [25]: a multi-agent communication method for limited bandwidth scenarios that uses a selecting mechanism for downsizing communication agents and learns the communication protocol with one real value message (float16 type). Additionally, we modify the MADDPG [36] and QMIX [41] algorithms to include communication as baselines to demonstrate that IMAC can be applied to various multi-agent methods and effectively handle limited bandwidth constraints.

3.6.1 Cooperative Navigation

In this particular scenario, a group of n agents must cooperate to reach k landmarks while avoiding collisions. These agents can observe the relative positions of

Algorithm 1: Informative Multi-agent Communication

```
Initialize the network parameters $\theta_a$, $\theta_{pro}$, $\theta_Q$, and $\phi_{sch}$
Initialize the target network parameters $\theta'_a$, and $\theta'_Q$
for $episode \leftarrow 1$ to $num\_episodes$ do
    Reset the environment for $t \leftarrow 1$ to $num\_step$ do
        Get features $h_i = [o_i, c_i]$ for each agent $i$
        Each agent $i$ gets messages from channel $m_i = \pi^i_{pro}(h_i)$
        Get scheduled message $c_i = f_{sch}(m_1, \cdots, m_n)$
        Each agent $i$ selects action based on features and messages
          $a_i = \pi^i_a(h_i, c_i)$
        Execute actions $a = (a_1, \cdots, a_n)$, and observe reward $r$ new
          observation $o_i$ for each agent $i$
        Store $(\boldsymbol{o_t}, a, r, \boldsymbol{o_{t+1}}, \boldsymbol{m}, \boldsymbol{c})$ in replay buffer $D$
        if $episode\%update\ threshold == 0$ then
            Sample a random mini-batch of $S$ samples $(\boldsymbol{o}, a, r, \boldsymbol{o'}, \boldsymbol{m}, \boldsymbol{c})$ from $D$
            Obtain the features $h'_i = [o'_i, c_i]$ and the messages $m_i$ for each agent
              $i$
            Set $y^j = r^j_i + \gamma Q^{\boldsymbol{\pi'}}_i(\boldsymbol{o}, a_1', \cdots, a_n')|_{a_k'=\pi^i_a{}'(h'_i, c_i)}$
            Update Critic by minimizing the loss
              $L(\theta) = \frac{1}{S}\sum_j (Q(\boldsymbol{o}, a_1, \cdots, a_n) - \hat{y})^2$
            Update policy, protocol and scheduler using the sampled policy
              gradients $\nabla_{\theta_i} \tilde{J}(\pi_i)$ for each agent $i$
            Update all target networks' parameters for each agent $i$:
              $\theta_i{}' = \tau\theta_i + (1-\tau)\theta_i{}'$
        end
    end
end
```

both other agents and landmarks, and they receive a shared reward based on the sum of distances between them and their nearest landmark. On the other hand, they receive a penalty for collisions with other agents. Throughout their learning process, agents must learn to identify and occupy the landmarks without colliding with others, using both their individual observations and information received from other agents.

To compare the performance of IMAC with other approaches, we selected TarMAC, GACML, and SchedNet, which represent the methods of learning the communication protocols via end-to-end training with the specific scheduler and clipping the messages. We also include a modified version of MADDPG with communication as a baseline, which was trained without the limited bandwidth constraint. This serves to show the baseline performance of the centralized training and decentralized execution.

Predator \Prey	IMAC	TarMAC	GACML	SchedNet	MADDPG w/ com
IMAC	**32.32\-4.26**	**28.91**\\ -22.27	**28.25** \\ -26.11	22.67 \\ -36.53	**34.33** \\ -22.62
TarMAC	25.13 \\ **-2.94**	23.45 \\ -20.42	22.12 \\ -16.51	**32.52** \\ -42.39	27.54 \\ -29.36
GACML	21.52 \**-12.74**	11.49 \\ -24.93	13.93 \\ -12.95	25.49 \\ -27.42	28.47 \\ -27.75
SchedNet	24.74 \**-9.63**	7.84 \\ -23.56	12.48 \\ -23.67	5.98 \\ -26.82	21.53 \\ -26.43
MADDPG w/ com	28.63 \\ -15.60	19.32 \\ -21.52	26.91 \\ -19.76	22.17 \\ -35.37	16.87 \\ **-13.09**

Table 3.1: Cross-comparison between IMAC and baselines on predator-prey.

In Figure 3.3 (a), we can observe the learning curve over 100,000 episodes, represented by the mean episode reward over a sliding window of 1000 episodes. At the end of the training, we observe that agents trained with communication achieve higher mean episode rewards. According to [23], an ”increase in reward when adding a communication channel” is sufficient for effective communication. Furthermore, IMAC outperforms other baselines throughout the training process, as it can reach the upper bound of performance early. The information bottleneck method used in IMAC allows for less redundant messages, which enables the agents to converge faster.

We also examine the performance of IMAC in scenarios with an increasing number of agents. To do so, we modify the environment such that each agent can only observe the nearest three agents and landmarks with relative positions and velocity, as suggested by [19]. Figure 3.3 (b) and (c) illustrate the superior performance of IMAC in the 5 and 10-agent scenarios.

Next, we evaluate the performance of IMAC and the modified MADDPG with communication under different limited bandwidth constraints during the execution stage. To achieve this, we first train IMAC with different prior distributions to satisfy different bandwidths, ensuring that the entropy of agents' messages satisfies the bandwidth constraints. We then constrain these algorithms to different bandwidths during the execution stage. Figure A.1 presents the density plot of episode reward per agent during the execution stage. We observe that IMAC with different prior distributions can achieve similar outcomes as MADDPG with communication, while MADDPG with communication fails in the limited bandwidth environment. Moreover, the same bandwidth constraint is less effective in IMAC than in MADDPG with communication, indicating that IMAC can discard useless information without impairing performance.

We also investigate the impact of limited bandwidth and β on multi-agent communication and the performance of agents. Figure 3.5 (a) demonstrates the

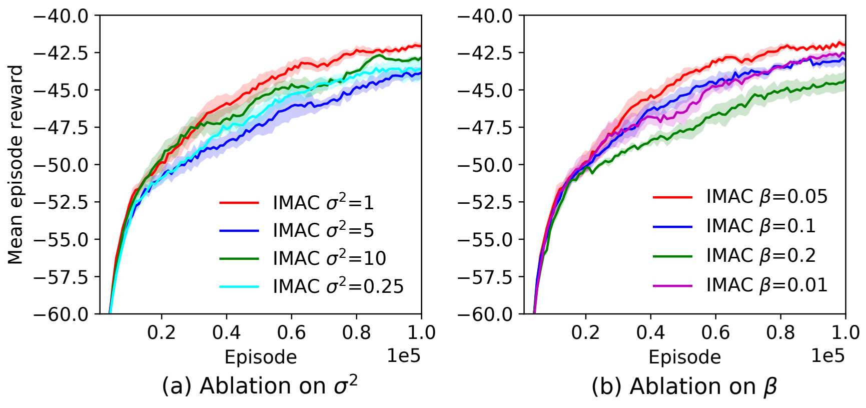


FIGURE 3.5: Ablation: learning curves with respect to Σ and β

learning curve of IMAC with different prior distributions, where IMAC with $z(M_i) = N(0,1)$ achieves the best performance. Smaller or larger variances lead to performance degradation, as smaller variances result in a more lossy compression, leading to less information sharing, while larger variances bring about more redundant information than the variance without regulation, leading to slower convergence. Figure 3.5 (b) shows a similar result for the ablation on limited bandwidth constraint, where larger β indicates a more strict compression and smaller β indicates a less strict one. Our ablation analysis indicates that the information bottleneck method extracts the most informative elements from the source, and a proper compression rate is crucial for multi-agent communication, as it can avoid losing much information caused by higher compression while resisting much noise caused by lower compression.

3.6.2 Predator Prey

In this scenario, there are m slower predators chasing n faster preys around an environment with l landmarks creating obstacles. Like in the cooperative navigation scenario, each agent observes the relative positions of other agents and landmarks. The predators share a common reward that is based on the collision between predators and preys, as well as the minimal distance between the two groups. The preys are penalized for running out of the screen boundary. This setup encourages the predators to learn to approach and surround the preys, while the preys learn to feint to save their teammates.

For this experiment, we set the number of predators to 4, the number of preys to 2, and the number of landmarks to 2. We use the same architecture as in the cooperative navigation scenario, and agents communicate only with their teammates. We train our agents through self-play for 100,000 episodes and then evaluate their performance by comparing IMAC with the baselines using cross-validation. We report the average episode rewards across 1000 rounds (episodes) as scores.

Predator \Prey	MADDPG_e1	MADDPG_e5	IMAC	IMAC_t5_e1	IMAC_t10_e1	IMAC_t10_e5
MADDPG_e1	18.01 \**-14.22**	24.15 \\-29.88	22.38 \\-16.91	47.59 \\-45.64	34.25 \\-27.68	50.81 \\-43.62
MADDPG_e5	26.32 \\-20.48	15.67 \**-11.59**	29.06 \\-22.16	27.07 \\-22.89	23.44 \\-20.41	32.24 \\-26.46
IMAC	**51.24** \\-42.56	**37.37** \\-45.521	**44.64** \\-36.49	**49.12** \\-42.65	**36.63** \\-30.03	35.42 \**-28.82**
IMAC_t5_e1	38.86 \\-32.06	34.54 \\-35.03	9.97 \**-3.11**	26.25 \\-21.06	11.80 \\-7.558	**38.32** \\-32.28
IMAC_t10_e1	26.67 \\-21.418	34.99 \\-35.02	9.71 \**-4.11**	9.82 \\-6.92	9.82 \\-6.92	37.50 \\-31.30
IMAC_t10_e5	45.88 \\-38.27	26.39 \\-35.42	11.51 \**-9.12**	30.02 \\-27.41	29.08 \\-25.661	22.25 \\-16.51

TABLE 3.2: Cross-comparison in different bandwidths on predator-prey. "t5" means that IMAC is trained with the variance $|\Sigma| = 5$. "e1" means that during the execution, we use the batch-norm like layer to clip the message to enforce its variance $|\Sigma| = 5$.

Comparison with baselines. For the predator-prey scenario, there are m slower predators and n faster preys in an environment with l landmarks. Similar to the cooperative navigation scenario, agents observe the relative position of other agents and landmarks. Predators are rewarded based on the collision with preys and the minimal distance between the two groups, while preys are penalized for running out of the boundary. In this setup, predators learn to approach and surround preys, while preys learn to avoid being caught.

The same baselines used in the cooperative navigation scenario are used for comparison. Table 3.1 shows the mean episode rewards of the predators and preys for IMAC and the baselines. Higher scores indicate better performance. The mean episode rewards of predators and preys show that IMAC outperforms the baselines in both roles, demonstrating better cooperation in competitive environments. The policy learned by IMAC predators and preys can also generalize to opponents with different policies.

Performance under stronger limited bandwidth. Similar to the cooperative navigation scenario, the algorithms are evaluated under different limited bandwidth constraints during execution. Table 3.2 shows the performance under different limited bandwidth constraints during inference for the predator-prey scenario. With limited bandwidth constraints, both MADDPG with communication and IMAC suffer from a degradation of performance. However, IMAC shows higher resistance to the effect of limited bandwidth and outperforms MADDPG with communication.

3.6.3 StarCraftII

We apply our method and baselines to decentralized StarCraft II micromanagement benchmark to show that IMAC can facilitate different multi-agent methods. We use the setup introduced by SMAC [40] and consider combat scenarios.

3m and 8m. Both tasks are symmetric battle scenarios, where marines controlled by the learned agents try to beat enemy units controlled by the built-in game AI. Agents will receive some positive (negative) rewards after having enemy (allied) units killed and/or a positive (negative) bonus for winning (losing) the battle.

Comparison with Baselines. To evaluate the effectiveness of IMAC, we compare it with QMIX with communication and QMIX with IMAC, as QMIX uses the centralized training decentralized execution scheme for discrete actions. We also evaluate MADDPG with communication, but it is designed for continuous control, while SMAC is a discrete-action scenario. Even if we modify MADDPG for discrete actions, it fails to get any positive reward. The learning curve of 200 episodes in terms of mean episode rewards is shown in Fig. 3.6. Initially, QMIX with IMAC has similar or even worse performance than QMIX with unlimited communication. However, as the training process continues, QMIX with IMAC performs better than QMIX with unlimited communication. This result suggests that IMAC can facilitate different multi-agent methods that have different centralized training schemes.

Performance under stronger limited bandwidth. We evaluate agents' performance under different limited bandwidth constraints. Results show a similar conclusion as in previous tasks (Details can be seen in the supplementary materials).

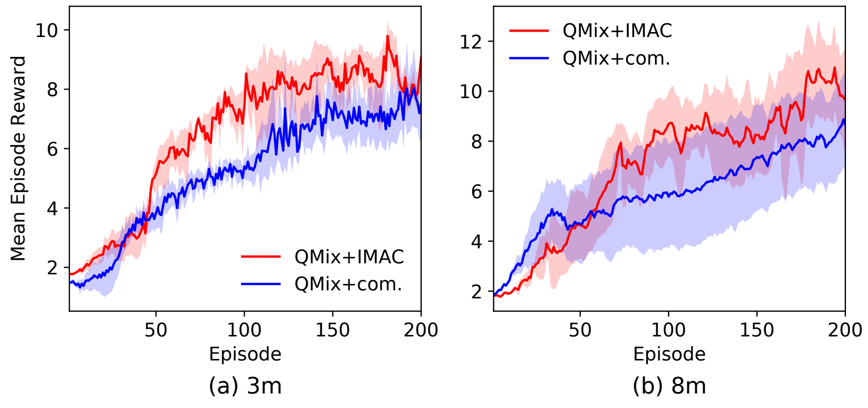


FIGURE 3.6: Learning curves comparing IMAC to other methods for 3m and 8m in Starcraft II.

3.7 Chapter Summary

In this chapter, we have proposed an *informative multi-agent communication* method in the limited bandwidth environment, where agents utilize the information bottleneck principle to learn an informative protocol as well as scheduling. We have given a well-defined explanation of the limited bandwidth constraint from the perspective of communication theory. We prove that limited bandwidth constrains the entropy of the messages. We introduce a customized batch-norm layer, which controls the messages'entropy to simulate the limited bandwidth constraint. Inspired by the information bottleneck method, our proposed IMAC algorithm learns informative protocols and a weight-based scheduler, which convey low-entropy and useful messages. Empirical results and an accompanying ablation study show that IMAC significantly improves the agents' performance under limited bandwidth constraint and leads to faster convergence.

Part II

Collaboration amongst agents

Chapter 4

I^2HRL: Interactive Influence-based Hierarchical Reinforcement Learning[1]

4.1 Introduction

Reinforcement Learning (RL) methods have recently yielded a plethora of positive results, including playing games like Go [43] and Atari [9], as well as controlling robots [13]. However, it is still challenging to learn policies in complex environments with large time horizons and sparse rewards. A promising method to address these issues is Hierarchical RL (HRL) that learns to operate at different temporal abstraction levels simultaneously. The need for HRL arises from the fact that in complex tasks, an agent may have to perform several sub-tasks in a particular order to achieve the final goal. However, these sub-tasks may themselves be complex and require several actions to be executed. In such cases, traditional RL methods may be insufficient as they require the agent to learn a single policy that can handle all sub-tasks and actions.

In HRL, the high-level policy is responsible for selecting subgoals and the low-level policy is responsible for executing actions to achieve those subgoals. The high-level policy operates at a slower time scale and selects subgoals based on the current

[1]The work in this chapter has been published in [42]

state of the environment. The low-level policy operates at a faster time scale and takes actions to achieve the subgoals selected by the high-level policy.

The collaboration between the high-level policy and low-level policy is crucial for the success of HRL. The high-level policy needs to select subgoals that are achievable by the low-level policy, and the low-level policy needs to execute actions to achieve those subgoals in the most efficient way possible.

The high-level policy and low-level policy communicate with each other through a set of shared variables, which are used to represent the current subgoal and its progress. The high-level policy sets the current subgoal, and the low-level policy uses this subgoal to guide its actions. The low-level policy updates the progress of the current subgoal, which the high-level policy uses to determine when to select a new subgoal.

The collaboration between the high-level policy and low-level policy in HRL can lead to more efficient and effective learning compared to traditional RL methods. By breaking down complex tasks into smaller subgoals, HRL can reduce the complexity of the learning problem and allow for more efficient exploration of the state space. Additionally, the high-level policy can provide guidance to the low-level policy, which can improve the speed and accuracy of learning.

However unfortunately, current HRL methods are subject to the non-stationarity problem [44, 45]. Namely, as the lower-level policy continues to change, a high-level action taken at the same state, but at different steps, may result in critically different state transitions and rewards. This negatively impacts policy exploration, since policies need more data at non-stationary states to stabilize training, i.e., an already visited non-stationary state may warrant further exploration. Yet, common exploration approaches in RL, such as count-based exploration [46], re-focus agents on less-visited states, which makes them ill-suited to support HRL. Though some remedies to HRL's non-stationarity issue were proposed, e.g., off-policy experience correction [44], or pre-training and freezing the low-level policy [47], they require additional manual configuration. This either breeds more hyperparameters or breaks the end-to-end scheme entirely.

To address these issues, we develop a novel HRL approach named *Interactive Influence-based Hierarchical Reinforcement Learning* (I^2HRL). Our contributions are threefold. First, we introduce a feedback loop from the process of low-level policy learning to the high-level policy. The latter can now condition its decisions on the features of the low-level behaviour policy. Second, we propose an influence-based framework and introduce information-theoretic regularization to control the dependency of the high-level policy on the changes in the low-level behaviour. This stabilises the high-level policy training. Finally, we propose an influence-based exploration method for the high-level policy that improves sample efficiency. Intuitively, if a state, where the dependency of the high-level policy on the low-level behaviour is stronger, is a potential failure point due to the changing low-level policy and should be explored more.

We compare our method with state-of-the-art HRL algorithms on several continuous controlling tasks in the MuJoCo domain [48]. Experimental results show that our method significantly outperforms existing algorithms. Bi-level communication/interaction with the influence-based framework can accelerate the learning process by 30%-50%, alleviate non-stationarity issues and improve data efficiency.

4.2 Related Work

HRL has been shown to be effective in dealing with long-horizon and sparse reward problems. Intuitively, HRL is (at least) a bi-level approach, where the high-level policy breaks down a problem into sub-tasks and learns to sequence them, while the low-level learns to resolve the sub-tasks efficiently. The specifics of the breakdown, and how exactly the high-level communicates to the low-level, varies among methods. The signal sent to the low-level can thus be some discrete values for selection of options [49] or skills [50], or it can be continuous vectors to set a subgoal in the state space [44] or latent space [51]. Generally, off-policy HRL algorithms [44, 45] are more efficient than on-policy algorithms [49, 51]. However, the off-policy scheme creates non-stationarity for the high-level policy, since the low-level policy is constantly changing. Various solutions to this issue were recently proposed, e.g., Hierarchical reinforcement learning with off-policy correction (HIRO) [44] uses joint training and an off-policy correction, which modifies old transitions into new ones that agree with the current low-level policy. Unfortunately, this means that the higher level may need to wait until the lower level policy converges before it can learn a meaningful policy itself. Hierarchical Actor-Critic (HAC) [45] introduces hindsight action transitions, which simulate a transition function using the optimal lower-level policy hierarchy, to train the high-level policy. Also, these transitions need carefully designed domain specific rewards. Other methods [47, 52, 53] break down the end-to-end manner by pre-training and fixing the low-level policy. The frozen low-level skills require a well-designed pre-training environment and cannot be adapted to all tasks. In contrast, we propose an efficient end-to-end solution by enabling bi-level communication among HRL levels. Furthermore, by regularizing the dependency of the high-level on the low-level policy we stabilize our method against non-stationarity.

Now, one of HRL promised benefits is structured exploration, i.e., explore with sub-task policies rather than primitive actions. However, the exploration/sample efficiency still matters [54]. Majority of HRL methods directly use single-agent exploration methods at the low-level, such as $\epsilon-$greedy or intrinsic motivation [55, 56]. Nonetheless, some works do focus on the high-level policy exploration, e.g., the diversity concept is often utilized to drive the variability in the high-level policy choices [47, 52]. The prerequisite, or course, is that a diverse set of low-level skills exists, which also requires well-designed pre-training environments. Moreover, the diversity guidance does not consider the interaction between the two levels. We notice that there is some work about the influence framework in multi-agent systems [57]. In this paper, we propose to guide the high-level exploration by the low-level policy influence. Intuitively, in a state, where the high-level action choice is less influenced by the particulars of the low-level policy, the high-level transition is near-stationary, and needs less exploration data.

Overall, our method can be best intuited if HRL is interpreted as a form of multi-agent reinforcement learning (MARL). After all, agent modeling and communication are feasible solutions for addressing non-stationarity in MARL [58]. By modelling the intentions and policies of others, agents can stabilize their training process. Training stabilization is also achieved through communication, where agents exchange information about their observations, actions and intentions. In our methods, considering the nature of cooperation between two levels of HRL, the low-level policy directly sends its policy representation to the high-level policy. To our knowledge, no prior work has connected these multi-agent solutions to HRL.

4.3 Preliminaries about Hierarchical Reinforcement Learning

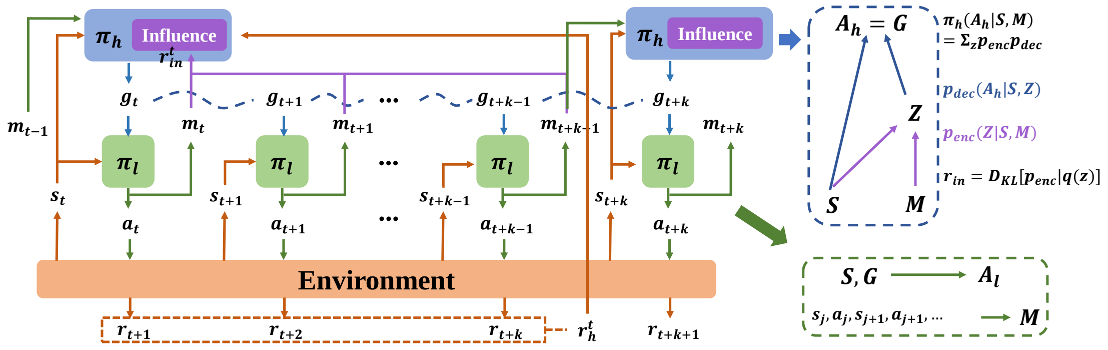


FIGURE 4.1: Overview of I²HRL, which consists of the interaction between the low-level and high-level policies, as well as an influence-based framework. At the high-level decision step t, the high-level policy receives the worker's previous message m_{t-1} and state s_t and sends a goal $g_t = \pi_h(s_t, m_{t-1})$. During the high-level decision interval $\{t+1, t+2, \cdots, t+k-1\}$, the subgoal follows the transition function $g_{t+i} = h(s_{t+i-1}, g_{t+i-1}, s_{t+i})$. The messages $\{m_t, m_{t+1}, \cdots, m_{t+k-1}\}$ sent by the low-level policy are used to compute the high-level intrinsic rewards. The rewards for the high-level policy: $r_h^t = \sum_{j=1,2,\cdots,k}(r(s_{t+j}) + r_{in}(s_{t+j}, m_{t+j}))$

One drawback of the generic RL is its inability to effectively handle long-term credit assignment problems, particularly in the presence of sparse rewards. HRL proposes methods for decomposing complex tasks into simpler subproblems that can be more readily tackled by low-level action primitives. We follow the two-level goal-conditioned off-policy hierarchy presented in HIRO [44]. A high-level policy π_h computes a state-space subgoal $g_t \sim \pi_h(s_t)$ every k time steps (g_t is also written as a_h in the rest of paper). Then a low-level policy π_l takes as an input the current state s_t and the assigned subgoal g_t and is encouraged to perform an action $a_t \sim \pi_l(s_t, g_t)$ that satisfies its subgoal via a low-level intrinsic reward function $r_l(s_t, g_t, s_{t+1})$. Finally, the high-level policy receives cumulative rewards $r_h = \sum_{i=1}^{k} r(s_{t+i})$. The low-level reward function is set as: $r_l(s_t, g_t, s_{t+1}) = -\|s_t + g_t - s_{t+1}\|_2$, and the subgoal-transition function is set as $g_{t+1} = \pi_h(s_t)$, if t mod $k = 0$, or otherwise using a fixed goal transition function $h(s_t, g_t, s_{t+1}) = s_t + g_t - s_{t+1}$.

4.4 Interactive Influence-based HRL

In this section, we present our framework for learning hierarchical policies with bi-direction communication and the high-level exploration. First, we make use of the low-level policy modeling and pass it to the high-level policy. In addition, we minimize the influence of the low-level policy which is quantified by the mutual information between the subgoals assigned by the high-level policy and the low-level policy representation. Lastly, we introduce the influence-based exploration with intrinsic rewards for the high-level policy.

We have two MDPs for the high-level policy and low-level policy respectively:

$$\mathrm{MDP}_h = (\mathcal{S}, \mathcal{A}_h, \mathcal{P}_h, r_h, \gamma, T/k)$$

$$\mathrm{MDP}_l = (\mathcal{S}, \mathcal{A}_l, \mathcal{P}_l, r_l, \gamma, k)$$

The non-stationarity emerges in the high-level policy transition. That is, at different steps, $\mathcal{P}_h$ outputs different probability and r_h outputs different rewards given the same transition. The cause of non-stationary transition functions is the change of the lower level policy. It is similar to the situation in the multi-agent reinforcement learning: the state transition function P and the reward function of each agent r_i depend on the actions of all agents. Each agent keeps changing to adapt to other agents, thus breaking the Markov assumption that governs most single-agent RL algorithms. It is natural to view HRL as cooperative two-agent reinforcement learning with such characteristics: (1) unshared rewards, (2) one-direction delayed communication and (3) fully observability.

4.4.1 Low-level Policy Modeling

Motivated by agent/opponent modeling, if the high-level policy can know or reason about the behaviors and the intentions of the low-level policy, the coordination efficiency will be improved and the training process of the agents might be stabilized. Due to the cooperation between the low-level and high-level policies, we propose the bi-direction communication where the low-level policy sends its policy representation to the high-level policy.

The low-level policy determines what to communicate. It is straight-forward to send the policy parameters $m = \theta_l$ as a complete representation of its own policy. However, it is well-known that the low-level policy network is often over-parameterized, which makes it hard to feed the parameters directly into the high-level policy. Also for agent modeling, one agent only needs to know the desires, beliefs, and intentions of others [59]. Consequently, it is practical to encode the low-level policy $m = f^m(\pi_l(\cdot\,,\cdot))$, where the f^m is an encoder function as the representation module.

In this work, we learn a representation function that maps episodes from the low-level policy π_l to a real-valued vector embedding. In practice, we optimize the parameters θ of a function $f^m_{\theta_m} : \mathcal{E} \to \mathbb{R}^d$ where $\mathcal{E}$ denotes the space of successive state-action transitions $\langle s^t, a_h^t, a_l^t, s^{t+1}, a_h^{t+1}, a_l^{t+1}, \cdots, s^{t+c}, a_h^{t+c}, a_l^{t+c}\rangle$ of size c corresponding to the low-level policy and d is the dimension of the embedding. We propose a principle for learning a good representation of a policy: *predictive representation*. The representation should be accurate for predicting the low-level policy actions given states.

For satisfying the principle, we utilize an imitation function via supervised learning. Supervised learning does not require direct access to the reward signal, making it an attractive task for reward-directed representation learning. Intuitively, the imitation function attempts to mimic the low-level policy based on the historical behaviours. Concretely, we utilize the representation function $f^m_{\theta_m} : \mathcal{E} \to \mathbb{R}^d$, as well as an imitation function $f_\phi : \mathcal{S} \times \mathcal{A} \times \mathbb{R}^d \to [0, 1]$ where ϕ are parameters of this imitation function that maps the low-level policy observation and embedding to a distribution over the agent's actions. We propose the following negative cross entropy objective to maximize with respect to ϕ and θ_m:

$$\mathop{\mathbb{E}}_{\substack{e_1 \sim D_l \setminus e_2 \\ \langle s, a_h, a_l\rangle \in e_2 \sim D_l}} \left[\log f_\phi \left(a_l | s, a_h, f^m_{\theta_m}(e_1)\right) \right] \tag{4.1}$$

where D_l is the replay buffer of the low-level policy. For the low-level policy, the objective function samples two distinct trajectories e_1 and e_2 (e_1 needs to be the history of e_2). The state-action pairs from e_1 are used to learn an embedding $f^m_{\theta_m}(e_1)$ that conditions the imitation function network trained on state-action pairs from e_2.

The imitation function is smaller than the low-level policy because it takes the low-level policy information as input. Also, it is trained more frequently than the low-level policy.

4.4.2 High-level Policy: Influence-based Framework

One may argue that the source of non-stationarity moves from the low-level policy to the low-level representation. That is, the high-level policy has the low-level policy representation but an out-of-date or inaccurate representation. We claim that the training process of the high-level policy is more stable with a representation than without a representation. Also, we introduce an influence-based framework for the high-level policy to extract useful information from the representation.

We expect the high-level policy to make decision with the most useful information and minimal influence of the low-level policy. In such situation, even if the low-level representation may be imprecise, the high-level policy can still make good decisions.

Consequently, we measure the influence via a conditional mutual information and regularize/minimize it for the high-level policy:

$$\begin{aligned} I(A_h; M|S) &= \sum_s \sum_m \sum_{a_h} p(a_h, m, s) \log \frac{p(s)p(a_h, m, s)}{p(a_h, s)p(m, s)} \\ &= \sum_s p(s) \sum_m p(m) \sum_{a_h} \pi(a_h|s, m) \log \frac{\pi_h(a_h|s, m)}{\pi_{h0}(a_h|s)} \\ &= \mathbb{E}_{\pi_h, \pi_l} \Big[D_{\mathrm{KL}} \left[\pi_h(a_h|s, m) | \pi_{h0}(a_h|s) \right] \Big] \end{aligned} \tag{4.2}$$

where A_h is the high-level action; M is the low-level policy representation; S is the state; $\pi_h(a_h|s, m)$ is the high-level policy. $\mathbb{E}_{\pi_h, \pi_l}$ is an abbreviation of $\mathbb{E}_{s,m,a_h \rangle \sim \pi_h, \pi_l}$ which denotes an expectation over the trajectories $\langle s, m, a_h \rangle$ generated by π_h, π_l. D_{KL} is the Kullback-Leibler divergence. $\pi_{h0}(a_h|s) = \sum_m p(m)\pi_h(a_h|s, m)$ is a *default* high-level policy without the low-level policy's influence, although the high-level policy never actually follows the *default* policy. $I(a_h; m|s) = 0$ if and only if a_h and m are independent given the state s. This mutual information can also be seen as a measure of stationarity. When the mutual information is small, whatever the low-level policy is, the high-level policy can always expect that a certain assignment of itself can lead to a high reward. The high-level policy's optimization objective is:

$$J(\theta_h) = \mathbb{E}_{\pi^h_{\theta_h},\pi^l}\Big[\sum_{i=0,k,\cdots}\gamma^i r_h^{t+i} - \beta D_{\mathrm{KL}}\left[\pi_h(a_h|s,m)|\pi_{h0}(a_h|s)\right]\Big] \tag{4.3}$$

We now focus on the second term because the optimization of the cumulative rewards term can be solved by standard RL algorithms. $\pi_h(a_h|s,m)$ is a deterministic policy, so we cannot directly compute the mutual information term. So we introduce an internal variable Z. Due to the data processing inequality (DPI) [37], $I(Z;M|S) \geq I(A_h;M|S)$. Therefore, minimizing $I(Z;M|S)$ also minimizes $I(A_h;M|S)$. We parameterize the policy $\pi_h(a_h|s,m)$ using an encoder $p_{\mathrm{enc}}(z|s,m)$ and a decoder $p_{\mathrm{dec}}(a_h|s,z)$ such that $\pi_h(a_h|s,m) = \sum_z p_{\mathrm{enc}}(z|s,m)p_{\mathrm{dec}}(a_h|s,z)$. Thus, we instead maximize this lower bound on $J(\theta_h)$:

$$\begin{aligned}
J(\theta_h) &\geq \mathbb{E}_{\pi^h_{\theta_h},\pi^l}\Big[\sum_{i=0,k,\cdots}\gamma^i r_h^{t+i}\Big] - \beta I(Z;M|S) \\
&= \mathbb{E}_{\pi^h_{\theta_h},\pi^l}\Big[\sum_{i=0,k,\cdots}\gamma^i r_h^{t+i} - \beta D_{\mathrm{KL}}\left[p_{\mathrm{enc}}(z|s,m)|p(z|s)\right]\Big]
\end{aligned} \tag{4.4}$$

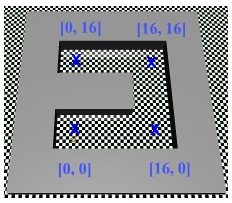
where $p(z|s) = \sum_m p(m)p_{\mathrm{enc}}(z|s,m)$ is the marginalized encoding. In practice, performing this marginalization over the representation may often be intractable since there may be a continuous distribution of representation.

Inspired by the information bottleneck principle and variation information bottleneck [30, 60], we use a Gaussian approximation $q(z|s)$ of the marginalized encoding $p(z|s)$. Since $D_{KL}[p(z|s)||q(z|s)] \geq 0$, that is, $\sum_z p(z|s)\log p(z|s) \geq \sum_z p(z|s)\log q(z|s)$, instead, we get its variational upper bound:

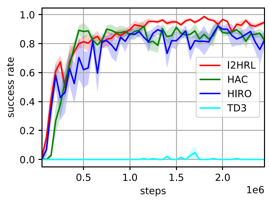
$$\begin{aligned}
I(Z;M|S) &= \mathbb{E}_{\pi^h_{\theta_h},\pi^l}\left[D_{\mathrm{KL}}[p_{\mathrm{enc}}(z|s,m)|p(z|s)]\right] \\
&= \sum_s p(s)\sum_m p(m)\sum_z p_{\mathrm{enc}}(z|s,m)\log p_{\mathrm{enc}}(z|s,m) \\
&\quad - \sum_s p(s)\sum_z p(z|s)\log p(z|s) \\
&\leq \sum_s p(s)\sum_m p(m)\sum_z p(z|m,s)\log\frac{p_{\mathrm{enc}}(z|s,m)}{q(z|s)} \\
&= \mathbb{E}_{\pi^h_{\theta_h},\pi^l}\left[D_{\mathrm{KL}}[p_{\mathrm{enc}}(z|s,m)|q(z|s)]\right]
\end{aligned} \tag{4.5}$$

This provides a lower bound of the objective in Eq. (4.3) $\tilde{J}(\theta_h) \leq J(\theta_h)$ that we maximize:

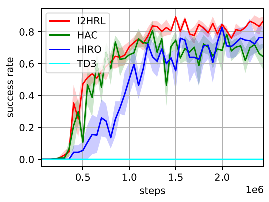
$$\tilde{J}(\theta_h) = \mathbb{E}_{\pi^h_{\theta_h}, \pi^l}\Big[\sum_{i=0,k,\cdots} \gamma^i r_h^{t+i} - \beta D_{\mathrm{KL}}\left[p_{\mathrm{enc}}(z|s,m)|q(z|s)\right]\Big] \tag{4.6}$$



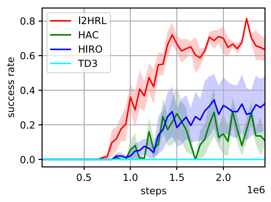
(A)



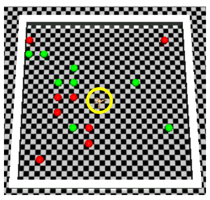
(B)



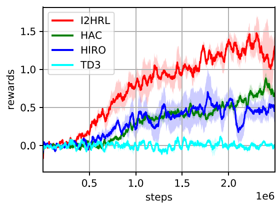
(C)



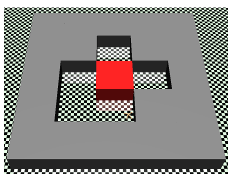
(D)



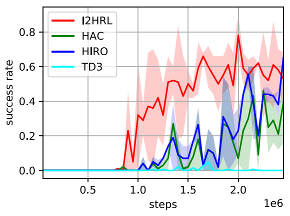
(E) A



(F)



(G)



(H)

Figure 4.2: Evaluation results in the AntMaze, AntGather, and AntPush. SR means success rate. (A) AntMaze Env. (B) SR in AntMaze [16,0] (C) SR in AntMaze [16,16] (D) SR in AntMaze [0,16] (E) ntGather Env. (F) Evaluation Reward in AntGather (G) AntPush Env. (H) Evaluation Reward in AntGather.

4.4.3 Influence-based Adaptive Exploration

Exploration in HRL heavily relies on the high level policy exploration [54]. The non-stationarity of high-level transition can lead to the inability of traditional exploration methods. During the training process of HRL, the non-stationary states instead of unvisited states need more data to stabilize the low-level policy. We can roughly split the HRL learning process into two phases with a vague boundary: (1) non-stationary phase; (2) near-stationary phase. In the first phase, i.e., at the beginning of the learning process, the low-level policy is always changing. Every state should be explored. In the second phase, the low-level policy is near-optimal. Most states are stationary for the high-level policy, thus the agent should pay more attention on the states which still cause non-stationarity. Process between these two phases can be seen as a mixture of both.

Consequently, we propose an influence-based reward for high level exploration, which is quantified by the KL divergence $D_{\mathrm{KL}}\left[p_{\mathrm{enc}}(z|s,m)|q(z)\right]$. Intuitively, if the KL divergence is small, the agent visits a state which is less easily influenced by the low-level policy. It means that the high-level transition is near-stationary, i.e., less data is needed here other than other states. When the KL divergence is large, a non-trivial influence of low-level policy is forced on the high-level policy. It means that the high-level policy needs more data to stabilize training.

During training, we utilize regularization mentioned in the previous section to train the high-level policy every k steps. Also, we calculate the KL divergence as the intrinsic reward every step with p_{enc} and $q(z)$ fixed:

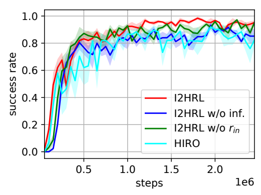
$$r_{in}^{t} = D_{\mathrm{KL}}\left[p_{\mathrm{enc}}(z^t|s^t, m^{t-1})|q(z^t)\right] \tag{4.7}$$

Eventually, instead of optimizing the Eq. (4.6), we derive the final objective for the high-level policy that we maximize:

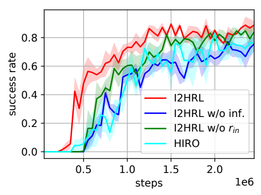
$$\tilde{J}'(\theta_h) = \mathbb{E}_{\pi^h_{\theta_h}}\Big[\sum_{i=0,k,\cdots}\gamma_h^i(r_h^{t+i} + \beta_r \sum_{j=0,1,\cdots,k} r_{in}^{j}) - \beta D_{\mathrm{KL}}\left[p_{\mathrm{enc}}(z|s,m)|q(z|s)\right]\Big] \tag{4.8}$$

where β_r is a factor.

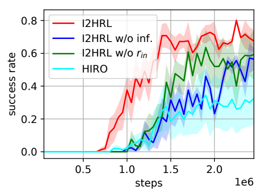
Even though the r_{in} and D_{KL} are in the same form, we notice that r_{in}, as the reward signal, will give credits to both $p_{\mathrm{enc}}(z|s,m)$ as well as $p_{\mathrm{dec}}(a_h|s,z)$, while D_{KL} is only used to update p_{enc}. Moreover, they are actually for adaptive exploration [61]. During the first phase, visited states at the early training steps can lead to similar high intrinsic rewards, thus contributing little to making different intrinsic rewards among states. Hence, the high-level policy explores a wide range of the state space. Meanwhile, the KL term makes effort to force the p_{enc} to decrease the influence of the low-level policy at these visited states, so that in the future, the intrinsic rewards here will be smaller and the states will be less visited. During the second phase, most states are stationary thus lead to similar low intrinsic rewards. However, once some states leads to non-stationarity and cause high intrinsic rewards, the overall policy will visit them more. It results in a narrow range of exploration with a focus on the non-stationary states.



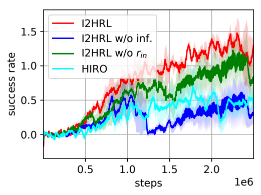
(A) AntMaze [16,0]



(B) AntMaze [16,16]



(C) AntMaze [0,16]



(D) AntGather

FIGURE 4.3: Ablation study for the proposed interaction, the influence-based framework and rewards. I^{2}HRL represents our method with Eq. (4.8); I^{2}HRL w/o inf represents that the internal variable z is removed, and the concatenation of the low-level policy representation with the state is feed into the high-level policy; I^{2}HRL w/o r_{in} represents the Eq. (4.6). HIRO is as a baseline.

4.5 Experiments

We design the experiments to answer the following questions: (1) How does I^{2}HRL compare against other end-to-end HRL algorithms? (2) Can influence-based framework stabilize the high-level policy training? (3) Is the proposed high-level exploration efficient under the non-stationrity?

4.5.1 Environmental Settings

We evaluate and analyze our methods in the benchmarking hierarchical tasks [48]. These environments were all simulated using the MujoCo physics engine for model-based control. The tasks are as follows:

Ant Gather. A quadrupedal ant is placed in a 20×20 space with 8 apples and 8 bombs. The agent receives a reward of $+1$ or collecting an apple and -1 for collecting a bomb; all other actions yield a reward of 0. Results are reported over the average of the past 100 episodes.

Ant Maze. The ant is placed in a U-shaped corridor and initialized at position $(0, 0)$. It is assigned an (x, y) goal-position that it is expected to reach (this (x, y) goal-position is only included in the state of the high-level policy). The agent is rewarded with its negative L_2 distance from current position to this goal position.

Ant Push. Th ant would move forward, unknowingly pushing the movable block until it blocks its path to the target. To successfully reach the target, the ant must first move to the left around the block and then push the block right, clearing the path towards the target location

We choose three methods as baselines:

- Twin Delayed Deep Deterministic Policy Gradient (TD3) [62]: a state-of-the-art flat RL algorithm in continuous control domain to validate the need for hierarchical models to solve these tasks.
- HIRO [44]: a state-of-the-art HRL algorithm which utilizes the off-policy correction to address the non-stationarity problem.
- HAC [45]: a state-of-the-art HRL algorithm which introduces hindsight action transitions to address the non-stationarity problem.

Results are reported over 5 random seeds of the simulator and the network initialization, and the time horizon is set to 500 steps. Both levels of I^{2}HRL utilize TD3. The low-level and high-level critic updates every single step and every 10 steps respectively. The low-level and high-level actor updates every 2 steps and every 20 steps respectively. We use Adam optimizer with learning rate of $3e-4$ for actor and critic of both levels of policies. We set the high-level policy decision interval k and the length of trajectories for low-level policy represent c as 10. Discount $\gamma = 0.99$, replay buffer size is $200,000$ for both levels of policies. The method-specific hyper-parameters (β and β_r) are fine-tuned for each tasks. All methods are well fine-tuned.

4.5.2 Results

Comparison with Baselines. In the Ant Gather, the results are the average external rewards of window size of 100. In the Ant Maze, the results are reported as success rates which are evaluated every 50,000 steps with the goal-position (16, 0), (16, 16), and (0, 16) respectively. The results, shown in Fig. 4.2 and 4.2f, demonstrate the training performance from scratch with I^{2}HRL and other baselines. In the Fig. 5.7c, all HRL methods reach a similar performance because it is easy for the ant to reach a goal which is directly in front of the ant. For the goal-position (16, 16) and (0, 16), agent is supposed to learn to change its direction. In the Fig. 5.7b, we can see that I^{2}HRL as well as HAC can reach the (16,16) earlier than HIRO, however, I^{2}HRL exceeds baselines by about 10% success rate when converged. In the Fig. 5.7a, which shows the success rate of algorithms in the most difficult task for ant, I^{2}HRL has a better performance than baselines on both the final performance and the convergence. In the AntGather and AntPush, I^{2}HRL also has a higher average rewards than other baselines. TD3 is non-hierarchical and not aimed for long-horizon sparse reward problems. Additionally, TD3 uses the Gaussian noise on the actions to explore, which serve as baseline exploration strategy. The success rates of TD3 in all maze tasks and the rewards in the gather task are almost zero. It shows the inability of flat RL algorithms on such tasks.

Ablations. To answer the question (2), we remove the internal variable z as well as the intrinsic reward, that is, directly feed the concatenation of the low-level policy representation and the state into the high-level policy. To answer the question (3), we only drop out the intrinsic reward r_{in}. In the Fig. 4.3, we notice that I^{2}HRL without influence-based framework has similar performance as compared with baseline. It demonstrates that the low-level policy representation can help the high-level policy alleviate the non-stationarity problem. However, we can also see that there are some drops and rebounds in both AntMaze and AntGather, like at the 2 million steps in Fig. (4.3c) and at 1.2 million steps in Fig. (4.3d). While I^{2}HRL without r_{in} does not have these drops and has better performance. As we mentioned in section 4.4.2, the source of non-stationarity moves from the low-level policy to the low-level representation. When the low-level policy representation is not sufficient or accurate to represent the low-level policy, the high-level policy makes bad decisions with bad representations. As the training process going, the representation becomes stable, thus the high-level policy starts to correct its understanding of the low-level representation. Also in the Fig. (4.3c), I^{2}HRL without r_{in} reaches a higher success rate from 1.5 million steps to 2 million steps than I^{2}HRL without the influence-based framework. It demonstrates the effectiveness of the proposed influence-based framework. The efficiency of proposed influence-based exploration is also shown in the Fig. (4.3). I^{2}HRL has better performance than I^{2}HRL without r_{in}, especially earlier to get the goal position and reach a higher success rate. It shows that r_{in} improves data efficiency, thus accelerating the learning process.

4.5.3 Visualization

Visualization of the low-level policy representation. We visualize the low-level policy representations in terms of different phases. We get the low-level policy representation (8 dimensions) every 100k steps over 3 million steps. Then we use t-SNE [63] to visualize the representations as shown in Fig. 4.4. At the beginning of the training process, the representation is sparse. While when the low-level policy is near-optimal, the representation is more dense.

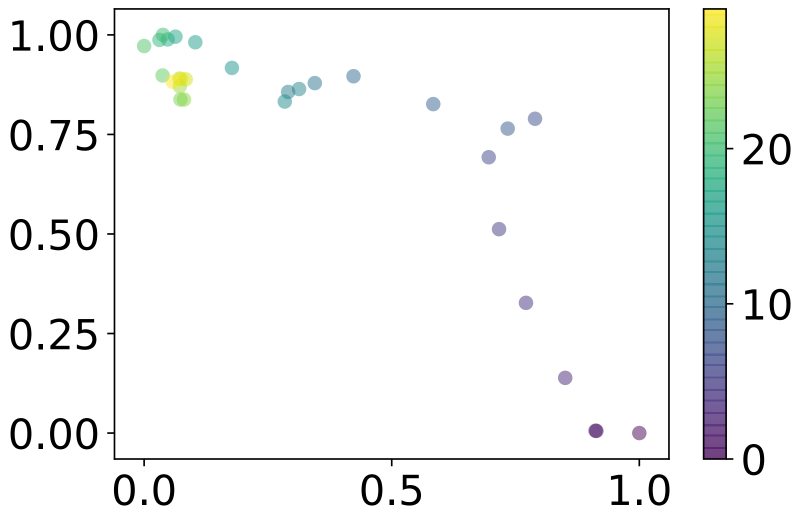
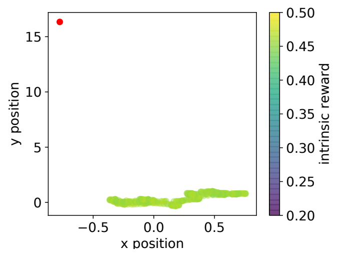
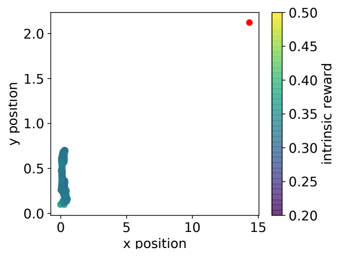


FIGURE 4.4: Visualization of the low-level policy representation.

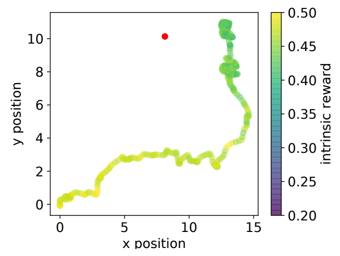
Visualization of non-stationary states. To backup our claims in Section 4.3, we visualize the intrinsic rewards in AntMaze with random target positions in terms of the different phases. Concretely, we split the training process into three phases: initial phase, non-stationary phase, and near-stationary phase. Since our method is end-to-end training, the intrinsic rewards are inaccurate in the initial phase. Consequently we pass the parameters of p_{enc} of the near-stationary phase to the parameters of the initial phase. At the beginning of the training process, the ant only navigates its surroundings with relatively high intrinsic rewards. When



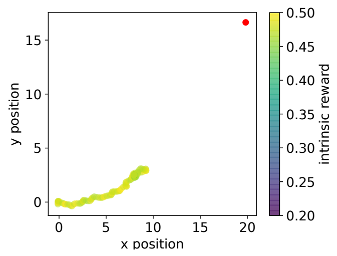
(A) initial phase



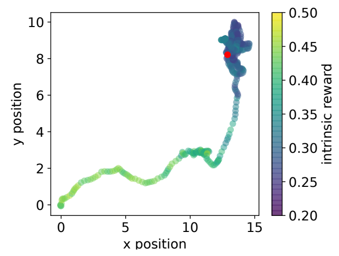
(B) initial phase



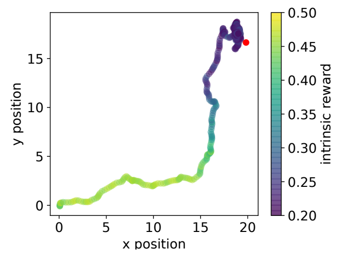
(C) non-stationary phase



(D) non-stationary phase



(E) near-stationary phase



(F) near-stationary phase

FIGURE 4.5: Visualization of the intrinsic reward. Axes represent the position of the ant. The red point is the goal position in AntMaze. Colormap shows the magnitude of the intrinsic rewards. (a)(b) The initial phase is at 0 step. (c)(d) The non-stationary phase is at 1 million steps. (e)(f) The near-stationary phase is at 3 million steps.

the low-level policy is near-optimal, the intrinsic rewards around goal positions are lower than rewards in the non-stationary phase.

As stated in Section 4.3, if the intrinsic reward is small, the agent visits a state which is less easily influenced by the low-level policy. It means that the high-level transition is near-stationary, i.e., less data is needed here other than other states. The result shows that once the ant learns how to approach a goal, the nearer it gets to the goal, the less data is needed. Because the task is goal-oriented, the ant actually needs to explore its surroundings when it does not know where to go. For example, a well-trained adventurer is walking in a desert, looking for an oasis. When he is far away from the oasis, he will walk around and explore to determine a direction. When he is approaching to the oasis, he will rush his goal without hesitation.

4.6 Chapter Summary

In this chapter, we focus on the non-stationarity problem in the end-to-end HRL. We propose the Interactive Influence-based HRL (I^{2}HRL). Concretely, we enable the interaction between the high-level policy and the low-level policy. Also, we propose the influence-based framework to address the non-stationarity. This framework also provides an influence-based adaptive exploration, which helps the agent to explore the more non-stationary states. We find that I^{2}HRL results in consistently better performance compared with state-of-the-art HRL baselines and accelerate the learning process.

I^{2}HRL splits the HRL into two parts: policy representation and communication. For the first one, more principles can be proposed to learn a good representation, such as predicting the low-level policy values (value-based), or predicting the state transitions and rewards (model-based). Also, various agent/opponent modeling methods can be future works to extend to learning the low-level policy representation. For the second part, the literature of multi-agent communication can also improve the cooperation between the two levels in HRL, such as centralized training or value decomposition.

Chapter 5

Commission Fee is not Enough: A Hierarchical Reinforced Framework for Portfolio Management[1]

5.1 Introduction

In this chapter, we consider the problem of portfolio management in finance with the application of HRL as a solution, especially from the perspective of the collaboration of the high-level policy and the low-level policy.

The problem of portfolio management is widely studied in the area of algorithmic trading. It aims to maximize the expected returns of multiple risky assets. Recently, reinforcement learning (RL) models are gaining popularity in the fintech community [65], with its great performance in different fields, including playing games [9, 43], controlling robots [13], and the Internet of things [66].

Due to the dynamic nature of markets and noisy financial data, RL has been proposed as a suitable candidate by its own nature [67, 68]. That is, RL can directly output trade positions, bypassing the explicit forecasting step by the trial-and-error interaction with the financial environment. Many existing RL methods get

[1]The work in this chapter has been published in Wang et al.

promising results by focusing on various technologies to extract richer representation, e.g., by model-based learning [69, 70], by adversarial learning [71], or by state augmentation [72]. However, these RL algorithms assume that portfolio weights can change immediately at the last price once an order is placed. This assumption leads to a non-negligible trading cost, that is, *the price slippage* – the relative deviation of the expected price of a trade and the price actually achieved. In a more realistic trading environment, the trading cost should be taken into consideration. Moreover, due to the need of balancing the long-term profit maximization and short-term trade execution, it is challenging for a single/flat RL algorithm to operate on different levels of temporal tasks.

In the field of finance, portfolio management is a crucial task that involves selecting and managing a collection of financial assets such as stocks, bonds, and mutual funds to meet the investment goals of an investor. It is a challenging problem due to the high dimensionality of the state and action spaces, as well as the complex dynamics and uncertainties of financial markets.

As our motivation, we observe that there is a hierarchy of portfolio managers and traders in real-world trading. Portfolio managers assign a percentage weighting to every stock in the portfolio periodically for a long-term profit, while traders care about the best execution at the favorable price to minimize the trading cost. This work gives the first attempt to leverage a similar idea to algorithmic trading. Concretely, we develop a **H**ierarchical **R**einforced trading system for **P**ortfolio **M**anagement (*HRPM*). Our system consists of a hierarchy of two decision processes. The high-level policy changes portfolio weights at a lower frequency, while the low-level policy decides at which price and what quantity to place the bid or ask orders within a short time window to fulfill goals from the high-level policy.

The collaboration of the high-level policy and the low-level policy is crucial in HRL-based portfolio management. The high-level policy provides guidance and goals to the low-level policy, while the low-level policy executes the investment decisions to achieve the goals. The high-level policy is responsible for setting the investment objectives, such as maximizing the expected return while controlling the risk, and providing abstract information about the state of the financial market to the low-level policy. The low-level policy, on the other hand, is responsible for selecting the specific financial assets to invest in based on the high-level policy's guidance and the current state of the market.

Through this collaboration, HRL-based portfolio management can achieve better performance and robustness than traditional portfolio management methods. The high-level policy can adapt to changes in the market conditions and investment goals, while the low-level policy can take advantage of the high-level policy's guidance to make more informed investment decisions. Overall, the use of HRL in portfolio management can provide investors with a more effective and efficient way to manage their portfolios.

Our contributions are four-fold. First, we formulate the portfolio management problem with trading cost as a hierarchical MDP, and combine portfolio management and trade execution by utilizing the hierarchical RL (HRL) framework to address the non-zero slippage. Second, as traditional HRL performs poorly in this problem, we propose the entropy bonus in the high-level policy to avoid risk, and use action branching to handle multi-dimensional action space. Third, to make full use of the stock data, we pre-train multiple low-level policies with an iterative scheme for different stocks in the limit order book environment, and then train a high-level policy on the top of the low-level policies. Fourth, we compare *HRPM* with baselines based on the historical price data and limit order data from the U.S. market and the China market. Extensive experimental results demonstrate that *HRPM* achieves significant improvement against state-of-the-art approaches. Furthermore, the ablation studies demonstrate the effectiveness of entropy bonuses and the trading cost is a non-negligible part in portfolio management.

5.2 Key Definitions in Portfolio Management

In this section, we introduce some definitions and the objective. Portfolio management is a fundamental financial task, where investors hold a number of financial assets, e.g., stocks, bonds, as well as cash, and reallocate them periodically to maximize future profit. We split the time into two types of periods: holding period and trading period as Fig. 5.1. During the holding period, the agent holds the pre-selected assets without making any purchase or selling. With the fluctuations of the market, the assets' prices would change during the holding period. At the end of the holding period, the agent will decide the new portfolio weights of the next holding period. In the trading period, the agent buys or sells some shares of assets to achieve the new portfolio weights. The lengths of the holding period and trading period are based on specific settings and can change over time.

Definition 5.1. (Portfolio) A portfolio can be represented as:

$$\boldsymbol{w}_t = [w_{0,t}, w_{1,t}, w_{2,t}, \dots, w_{M,t}]^T \in \mathbb{R}^{M+1} \text{ and } \sum_{i=0}^{M} w_{i,t} = 1 \tag{5.1}$$

where $M+1$ is the number of portfolio's constituents, including one risk-free asset, i.e., cash, and M risky assets, i.e., stock[2]. $w_{i,t}$ represents the ratio of the total portfolio value (money) invested at the beginning of the holding period t on asset i. Specifically, $w_{0,t}$ represents the cash in hand. Also, we define $\boldsymbol{w'_t}$ as the portfolio weights at the end of the holding period t.

Definition 5.2. (Asset Price) The opening prices $\boldsymbol{p}_t$ and the closing prices $\boldsymbol{p}'_t$ of all assets for the holding period t are defined respectively as $\boldsymbol{p}_t = [p_{0,t}, p_{1,t}, p_{2,t}, \dots, p_{M,t}]^T$ and $\boldsymbol{p}'_t = \left[p'_{0,t}, p'_{1,t}, p'_{2,t}, \dots, p'_{M,t}\right]^T$. Note that the price of cash is constant, i.e., $p_{0,t} = p'_{0,t} = p_{0,t+1}$ for $t \geq 0$.

Definition 5.3. (Portfolio Value) We define v_t and v'_t as portfolio value at the beginning and end of the holding period t. So we can get the change of portfolio value during the holding period t and the change of portfolio weights:

$$v'_t = v_t \sum_{i=0}^{M} \frac{w_{i,t} p'_{i,t}}{p_{i,t}} \text{ and } w'_{i,t} = \frac{\frac{w_{i,t} p'_{i,t}}{p_{i,t}}}{\sum_{i=0}^{M} \frac{w_{i,t} p'_{i,t}}{p_{i,t}}} \text{ for } i \in [0, M] \tag{5.2}$$

Definition 5.4. (Market order) A market order refers to an attempt to buy or sell a stock at the current market price, expressing the desire to buy (or sell) at the best available price.

Definition 5.5. (Limit Order) A limit order is an order placed to buy or sell a number of shares at a specified price during a specified time frame. It can be modeled as a tuple $(p_{target}, \pm q_{target})$, where p_{target} represents the submitted target price, q_{target} represents the submitted target quantity, and $\pm$ represents trading direction (buy/sell).

Definition 5.6. (Limit Order Book) A limit order book (LOB) is a list containing all the information about the current limit orders.

[2]In this chapter, we focus on the stocks for ease of explanation. This framework is applicable to other kinds of assets.

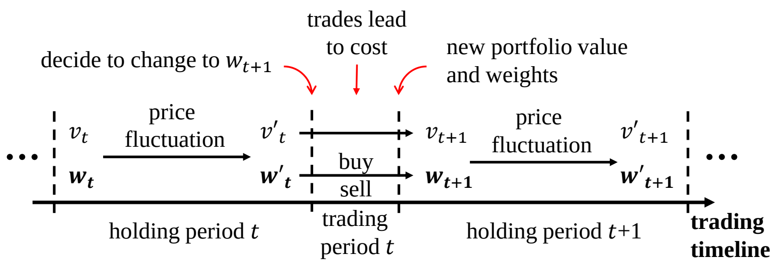


FIGURE 5.1: Illustration of the portfolio management process.

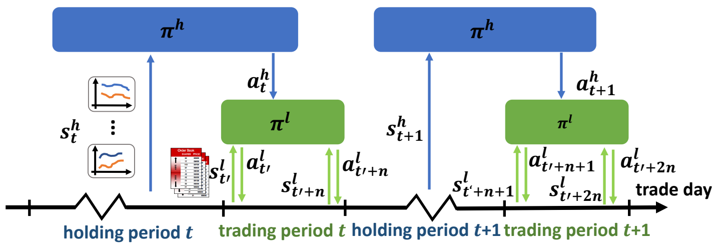
We observe that a strategy pursuing a better selling price is always associated with a longer (expected) time-to-execution. However, the long time-to-execution could be very costly, since a strategy might have to trade at a bad price at the end of the trading period, especially when the market price moves downward.

Definition 5.7. (Trading Cost) Trading cost consists of commission fee and slippage: $c_{trade} = c_{com} + c_{slippage}$. Commissions are fees that brokers charge to implement trades and computed as $c_{com} = \lambda \sum_{i=1}^{M} (q_{i,target} \times p_{i,avg})$, where λ is the constant commission rate for both selling and buying. Slippage is the difference between the expected price of a trade and the price actually achieved. Here we define the slippage as the average execution price p_{avg} achieved by the strategy relative to the price at the end of the trading period t: $c_{slippage} = (p_{avg} - p_{t+1}) \times (\pm q_{target})$.

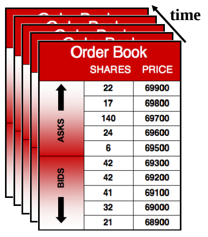
Considering the trading cost, the change of portfolio value during a holding period and a trading period t satisfies:

$$\begin{aligned} v_{t+1} = & \underbrace{v_t w_{0,t} - \sum_{i=1}^{M} [\pm q_{i,target} \times p_{i,avg} + \lambda q_{i,target} \times p_{i,avg}]}_{\text{Cash}} \\ & + \underbrace{\sum_{i=1}^{M} \left[(\frac{v_t w_{i,t}}{p_{i,t}} \pm q_{i,target}) \times p_{i,t+1} \right]}_{\text{Assets Value}} \\ = & \sum_{i=0}^{M} \frac{v_t w_{i,t} p_{i,t+1}}{p_{i,t}} - c_{trade,t} \end{aligned} \tag{5.3}$$

Our objective is to maximize the final portfolio value given a long time horizon by taking into account the trading cost.



(A) *HRPM* Model



(B) The low-level states

FIGURE 5.2: Overview of *HRPM*

5.3 Hierarchical MDP Framework for Portfolio Management

A key challenge to the real-world portfolio management is to balance the multi-faceted and sometimes conflicting objectives of different decision processes. Portfolio managers' main concerns are about longer-term profit. They typically think of execution as a chore to fulfill the clients' profit requirements and risk preferences. While traders care about best execution at the most favorable price to minimize the trading cost, but might not always be aware of managers' sources of profit and urgency of trade. Moreover, flat RL cannot handle complex tasks with long time horizons. Portfolio managers take many hours (sometimes days) while agents need to make decisions every few seconds or faster. The sampling frequency is limited by this time horizon issue so that all available information about market dynamics is not fully utilized. Consequently, electronic trading algorithms are supposed to operate on multiple levels of granularity.

Formally, we model the portfolio management with trading cost as two hierarchical MDPs, respectively for portfolio management and trade execution:

$$\text{MDP}^h = (\mathcal{S}^h, \mathcal{A}^h, \mathcal{P}^h, r^h, \gamma, T^h)$$

$$\text{MDP}^l = (\mathcal{S}^l, \mathcal{A}^l, \mathcal{P}^l, r^l, \gamma, T^l)$$

where the detailed definitions are given in the following subsections. The high-level policy and the low-level policy are executed in different timescales. Concretely, the time horizon for the high-level MDP is the total holding time of the portfolio, and the timestep of the high-level MDP is the holding period, which might consist of several trading days. On the other hand, the time horizon for the low-level MDP is a small trading window, e.g., a trading day. The timestep of the low-level MDP could be several minutes for the low-level policy to take actions. We use t as the high-level timestep, and t' as the low-level timestep.

5.3.1 The High-level MDP for Portfolio Management

The high-level MDP is modeled for the portfolio management problem. That is, the trading agent gives new portfolio weights for reallocating the fund into a number of financial assets at the beginning of each holding period.

State. We describe the state $s_t^h = \{\boldsymbol{X}_t, \boldsymbol{w_t}\} \in \mathcal{S}^h$ with historical stock basic features X_t, and the current portfolio weights $\boldsymbol{w_t}$. For the M non-cash assets, the features $\boldsymbol{X}_{i,t}$ for asset i are built by a time window k: $\boldsymbol{X}_{i,t} = \{\boldsymbol{x}_{i,t-k+1}, \boldsymbol{x}_{i,t-k+2}, \cdots, \boldsymbol{x}_{i,t}\}$, where $\boldsymbol{x}_{i,t}$ presents the basic information of asset i at trading day t, including opening, closing, highest, lowest prices and volume. Note that the state can be enriched by adding more factors for better performance. We only use basic daily information to describe high-level state in this work for simplification and fair comparison.

Action. At the end of the holding period t, the agent will give a high-level action a_t^h based on the state s_t^h to redistribute the wealth among the assets as a subtask. The subtask is determined by the difference between portfolio weights $\boldsymbol{w_t'}$ and $\boldsymbol{w_{t+1}}$. Since $\boldsymbol{w_t'}$ has already been determined in the holding period t according to Eq. (5.2), the action of the agent in holding period t can be represented solely by the portfolio vector $\boldsymbol{w_{t+1}}$. Consequently, we define the action as $a_t^h = \boldsymbol{w_{t+1}}$ in the continuous action space, and the action a_t holds the properties in Eq. (5.1).

Reward. For each state-action pair, i.e., (s_t^h, a_t^h) at holding period t, according to Eq. (5.3), $r_t = v_{t+1} - v_t$.

5.3.2 The Low-level MDP for Trade Execution

The high-level policy assigns subtasks for the low-level policy. The subtasks are represented by an allowed time window with length of T_{window} and the target trading quantity q_{target}. T_{window} is based on manual configurations, and $q_{target,t}$ of trading period t can be computed by $q_{target,t} = v'_t|w_{t+1} - w'_t| \odot \frac{1}{p'_t}$, with assumption of a continuous market. The subtasks can be considered as the trade execution, where the trading agent places many small-sized limit orders at different times at corresponding desired prices. As a result, we model the subtask as the low-level MDP.

State. We maintain a state $s^l_{t'}$ at each low-level time step t', considering the private states and the market states. The private states of trade execution are remaining trading time $T_{window} - T_{elapsed}$ and the remaining quantity $q_{target,t} - q_{finished}$, where $T_{elapsed}$ is the used time and $q_{finished}$ is the finished quantity. The market states consist of the historical limit order books and multiple technical factors as shown in Fig. 5.2b. Specifically, we collect historical LOBs in a time window k': $\boldsymbol{LOB}_{t'} = \{\boldsymbol{lob}_{t'-k}, \boldsymbol{lob}_{t'-k+1}, \cdots, \boldsymbol{lob}_{t'-1}\}$, where $\boldsymbol{lob}_{t'}$ is the limit order book at time step t', including price and volume.

Action. Each low-level action corresponds to a limit order decision with the target price p_{target} and the target quantity $\pm q_{target}$. Note that a zero quantity indicates that we skip the current trading time step with no order placement, and if an action $a_{t'}$ fails to be executed in the low-level time step t' due to an inappropriate price, the action will expire at the next low-level time step $(t' + 1)$ without actual trading. Any quantity remaining at the end of the trading period must be cleaned up by using a market order, walking through the lower prices on the bid side of the order book until all remaining volumes are sold.

Reward. Once receiving a limit order decision, the environment will match this order and feedback an executed price p_{paid}. We define the low-level reward as: $r^l_{t'} = -\left[\lambda(\pm q_{target} \times p_{paid}) + (p_{paid} - p_{t+1}) \times (\pm q_{target})\right]$. So the cumulative trading cost in a low-level episode will be reported to the high-level policy to compute the high-level reward $c_{trade,t} = -\sum_{t'}^{T^l} r^l_{t'}$.

5.4 Optimizing via Deep Hierarchical Reinforcement Learning

In this section, we first present our solution for learning hierarchical policies, then propose our solution algorithms respectively for the two levels of MDPs. Finally, we introduce two schemes: pre-training and iterative training for better financial data usage.

5.4.1 Hierarchy of Two Policies

Our environment has two dynamics: the price dynamic and the limit order book dynamic. We extend the standard RL setup to a hierarchical two-layer structure, with a high-level policy π^h and a low-level policy π^l. Different from the existing HRL algorithms, our high-level policy and the low-level policy run in different timescales. As shown in Fig. 5.2a, each high-level time step t consists of three steps:

1. The higher-level policy observes the high-level state s_t^h and produces a high-level action a_t^h, which corresponds to the new portfolio for the next step.
2. The high-level action a_t^h would generate subtasks for the low-level policy according to the gap between the current portfolio and the new one.
3. The low-level policy produces the low-level action $a_{t'}^l$ based on the low-level states $s_{t'}^l$. These actions would be applied to the limit order dynamic for updating the current portfolio.

5.4.2 High-level RL with Entropy Bonus

We consider the high-level problem as a continuous control problem, since the high-level policy is supposed to generate a continuous vector according to Eq. (5.1). Previous works utilize DDPG [13] to generate portfolio weights. Unfortunately, DDPG faces challenges in solving our problem. First, DDPG highly relies on a well-trained critic. However, the critic also needs lots of data, which is limited in the financial area [71]. Additionally, the exploration in DDPG is dependant on the Gaussian noise when applying actions into the environment. This exploration might fail due to choosing specific sequences of portfolio weights and excluding other possible sequences of portfolio weights. Also, we find that the portfolio weights will converge to a few assets. That is, the agent only holds few assets.

Here we utilized the REINFORCE algorithm [73]. One approach to address these issues is to introduce an entropy bonus. In order to encourage the high-level policy not to "put all the eggs in one basket", we aim to find a high-level policy that maximizes the maximum entropy objective:

$$\begin{aligned}\pi^{h*} &= \underset{\pi^h}{\operatorname{argmax}} \sum_{t=0}^{T^h} E_{s_0^h \sim \beta_0, s_{t+1}^h \sim p^h, a_t^h \sim \pi^h} [\gamma^h \tilde{r}^h(s_t^h, a_t^h)] \\ &= \underset{\pi^h}{\operatorname{argmax}} \sum_{t=0}^{T^h} E_{s_0^h, s_{t+1}^h, a_t^h} \left[\gamma^h \left(r^h(s_t^h, a_t^h) + \eta \mathcal{H}(a_t^h)\right)\right]\end{aligned} \tag{5.4}$$

where β_0 is the intial high-level state distribution, $s_{t+1}^h \sim p^h$ is an abbreviation of $s_{t+1}^h \sim p^h(\cdot|s_t^h, a_t^h)$, $a_t^h \sim \pi^h$ is an abbreviation of $a_t^h \sim \pi^h(\cdot|s_t^h)$, η determines the relative importance of the entropy term versus the reward, $\mathcal{H}(a_t^h)$ is the entropy of the portfolio weights, which is calculated as $\mathcal{H}(a_t^h) = \mathcal{H}(\boldsymbol{w_{t+1}^h}) = -\sum_i w_{i,t+1}^h \log w_{i,t+1}^h$, and $\tilde{r}^h$ is an abbreviation of the summation of the high-level reward and entropy bonus.

We use the softmax policy as the high-level policy $\pi^h(s^h, a^h; \theta) = \frac{e^{\theta^T \cdot \phi(s^h, a^h)}}{\sum_b e^{\theta^T \cdot \phi(s^h, b)}}$ for all $s^h \in \mathcal{S}^h, a^h \in \mathcal{A}^h$. Let $J(\pi_\theta^h) = \mathbb{E}_{\pi^h}\left[\sum_{t=0}^T \gamma^t t_t\right]$ denote the expected finite-horizon discounted return of the policy. According to the policy gradient theorem [73], we compute the gradient $\nabla_\theta J\left(\pi_\theta^h\right) = \underset{\tau \sim \pi_\theta}{\mathrm{E}}\left[\sum_{t=0}^T \nabla_\theta \log \pi_\theta^h\left(a_t|s_t\right) G_t\right]$, where $G_t = \sum_{j=0}^{T-t} \gamma^j \tilde{r}^h$. Thus we update the high-level policy parameters by $\theta \leftarrow \theta + \alpha \gamma^t \nabla_\theta \log \pi_\theta^h\left(a_t|s_t\right) G_t$.

5.4.3 Low-level RL with Action Branching

The low-level problem is considered as a discrete control problem with two action dimensions: price and quantity. We utilize the Branching Dueling Q-Network [74]. Formally, we have two action dimensions with $|p^l| = n_p$ discrete relative price levels and $|q^l| = n_q$ discrete quantity proportions. The action value Q_d^l at state $s^l \in \mathcal{S}^l$ and the action $a_d^l \in \mathcal{A}_d^l$ are expressed in terms of the common state value $V^l(s)$ and the corresponding (state-dependent) action advantage $Adv_d^l\left(s, a_d^l\right)$ for $d \in \{p, q\}$:

$$Q_d^l(s^l, a_d^l) = V(s^l) + (Adv_d(s^l, a_d^l) - \frac{1}{n} \sum_{a_d^{l\prime} \in \mathcal{A}_d} Adv_d(s^l, a_d^{l\prime}))$$

We train our Q-value function based on the one-step temporal-difference learning:

$$y_d = r + \gamma Q_d^-(s^{l'}, \underset{a_d^{l\,\prime} \in \mathcal{A}_d}{\arg\max} Q_d(s^{l'}, a_d^{l\,\prime})), d \in \{p, q\} \tag{5.5}$$

$$L = \mathbb{E}_{(s,a,r,s') \sim \mathcal{D}}[\frac{1}{N} \sum_{d \in \{P,I\}} (y_d - Q_d(s^l, a_d^l))^2] \tag{5.6}$$

where $\mathcal{D}$ denotes a (prioritized) experience replay buffer and a denotes the joint-action tuple $\left(p^l, q^l\right)$.

5.4.4 Training Scheme

The financial data, including price information, multiple factors, and limit order book, is complex and high-dimensional. In order to use the available data more efficiently, we utilize the pre-training scheme and the iterative training.

Pre-training. In this work, we pre-train the low-level policy in the limit order book environment. Specifically, we first pre-select several assets, i.e., stocks, as our pre-trained data sources. By individually training the corresponding buying/selling policy of each asset, we derive multiple low-level policy parameters for these assets. In the general setting of hierarchical reinforcement learning, the high-level policy and the low-level policy are only trained together in a single environment. However, this training scheme suffers from data insufficiency, since the agent's interactions with the environment are severely restricted due to the joint training, especially for the low-level policy training. By introducing the pre-training scheme, the low-level policy would be trained with more and diverse interactions, thereby having better generalization and robustness.

Iterative training. Following the iterative training scheme in [75], we augment the private state repeatedly in the low-level pre-training. Specifically, we traverse the target quantity from $(0, 0)$ to a max target quantity q_{max} and the remaining time from 0 to a max trading time window T'_{max}. In this way, the low-level policy is trained with the augmented private states, thus can generalize different subtasks assigned by the high-level policy.

	the U.S. Market	*the China Market*
Training	2000/01/03-2014/01/03	2007/02/08-2015/07/03
Evaluate	2014/01/06-2015/12/31	2015/07/06-2017/07/03
Test	2016/01/04-2018/05/22	2017/07/04-2018/07/04

TABLE 5.1: Period of stock data used in the experiments.

5.5 Experiment

In this section, we introduce our data processing, baselines, and metrics. Then we evaluate our algorithm with baselines in different markets. Lastly, we analyze the impact of entropy in reward and the trading cost by ablation study.

5.5.1 Dataset Setting and Preprocessing

Our experiments are conducted on stock data from the U.S. stock market and China stock market. The stock data are collected from wind[3]. We evaluate our algorithm on different stock markets to demonstrate robustness and practicality. We choose stocks with large volumes so that our trading action would not affect the market price. Specifically, we choose 23 stocks from *Dow Jones Industrial Average Index (DJIA)* and 23 stocks from *SSE 50 Index* for the U.S. stock market and China stock market respectively. In addition, we introduce 1 cash asset as a risk-free choice for both of the two markets. Moreover, the period of stock data used in the experiments is shown in Table 5.1.

For the high-level setting, we set the holding period to 5 days and set the trading period to one day. We set the time window of the high-level state as 10 days. We normalize the stock data to derive a general agent for different stocks in data pre-processing. Specifically, we divide the opening price, closing price, high price, and low price by the close price on the first day of the period to avoid the leakage of future information. Similarly, we use the volume on the first day to normalize the volume. For missing data which occurs during weekends and holidays, we fill the empty price data with the close price on the previous day and set zero volume to maintain the consistency of time series . For the constant commission rate, we set it to 0.2% on both two markets.

[3]https://www.wind.com.cn/

For the low-level setting, we set the time interval to 30 seconds. We set the time window of the low-level state as opening hours of one day. Similarly, we also normalize the prices in limit order books by dividing the first price at the beginning of the period, as well as the volume. For missing data which occurs during weekends and holidays, we fill the empty price data with the previous price and set zero volume to maintain the consistency of time series.

5.5.2 Baseline and Metrics

Baselines. We compare our methods with the baselines:

- *Uniform Constant Rebalanced Portfolios (UCRP)* [76] keeps the same distribution of wealth among a set of assets from day to day.
- *On-Line Moving Average Reversion (OLMAR)* [77] exploits moving average reversion to overcome the limitation of single period mean reversion assumption.
- *Weighted Moving Average Mean Reversion (WMAMR)* [78] exploits historical price information by using equal weighted moving averages and then learns portfolios by online learning techniques.
- *Follow the Winner (Winner)* [79] always focus on the outperforming asset and transfers all portfolio weights to it.
- *Follow the Loser (Loser)* [80] moves portfolio weights from the outperforming assets to the underperforming assets.
- *Deep Portfolio Management(DPM)* [81] is based on Ensemble of Identical Independent Evaluators (EIIE) topology. *DPM* uses asset prices as state and trains an agent with a Deep Neural Network (DNN) approximated policy function to evaluate each asset's potential growth in the immediate future and it is the state-of-the-art RL algorithm for PM.

In addition, *SSE 50 Index* and *Dow Jones Industrial Average Index (DJIA)* are introduced as baselines in the comparison of performance to demonstrate whether these portfolio strategies are able to outperform the market.

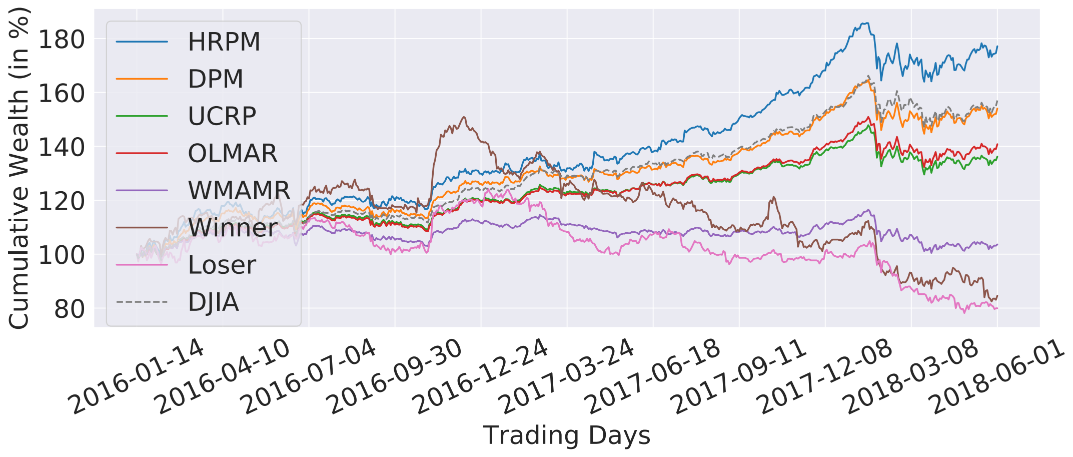


FIGURE 5.3: The portfolio value in the U.S. market.

	ARR(%)	ASR	MDD	DDR
UCRP	13.827	1.24	0.124	1.348
Winner	-6.773	-0.474	0.454	-0.455
Loser	-8.961	-0.911	0.371	-0.723
OLMAR	15.428	1.252	0.123	1.503
WMAMR	1.503	0.468	0.138	0.143
DPM	19.864	**1.263**	0.121	1.815
HRPM	**27.089**	1.246	**0.117**	**2.403**
DJIA	*20.823*	*1.206*	*0.116*	*2.547*

TABLE 5.2: Performance comparison in the U.S. market

Metrics. we use the following performance metrics for evaluations: *Annual Rate of Return (ARR)*, *Annualized Sharpe Ratio (ASR)*, *Maximum DrawDown (MDD)* and *Downside Deviation Ratio (DDR)*.

- *Annual Rate of Return (ARR)* is an annualized average of return rate. It is defined as $ARR = \frac{V_f - V_i}{V_i} \times \frac{T_{year}}{T_{all}}$, where V_f is final portfolio value, V_i is initial portfolio value, T_{all} is the total number of trading days, T_{year} is the number of trading days in one year.
- *Annualized Sharpe Ratio (ASR)* is defined as *Annual Rate of Return* divided by the standard deviation of the investment. It represents the additional amount of return that an investor receives per unit of increase in risk.

	ARR(%)	ASR	MDD	DDR
UCRP	23.202	2.917	0.136	1.863
Winner	-13.6	-1.201	0.368	-0.705
Loser	-17.645	-2.1	0.293	-1.045
OLMAR	24.947	3.252	0.122	1.961
WMAMR	15.873	2.485	0.137	2.775
DPM	35.81	3.126	0.153	2.404
HRPM	**48.742**	**3.813**	**0.131**	**3.325**
SSE50	*6.081*	*1.1*	*0.172*	*0.641*

TABLE 5.3: Performance comparison in the China market

- *Maximum DrawDown (MDD)* is the maximum observed loss from a peak to a bottom of a portfolio, before a new peak is attained. It is an indicator of downside risk over a specified time period.
- *Downside Deviation Ratio (DDR)* measures the downside risk of a strategy as the average of returns when it falls below a *minimum acceptable return (MAR)*. It is the risk-adjusted ARR based on Downside Deviation. In our experiment, the *MAR* is set to zero.

5.5.3 Comparison with Baselines

Backtest Results in the U.S. Market. Figure 5.3 shows the cumulative wealth vs. trading days in the U.S. market. From the plots of *DJIA* index, we can see this period is in a bull market generally, although the market edges down several times. While simple methods like *Following the Winner* and *Following the Loser* fail, the portfolio values conducted by most of the strategies keep climbing but still underperform the market. Particularly, our *HRPM* is the only strategy that outperforms the *DJIA* index, when *DPM* keeps almost the same as the trend of the market.

We can compare the metrics of different strategies in detail in Table 5.4. On *ARR*, *HRPM* performs much better than the other methods and outperforms the market with 6.2%. On *ASR*, most of methods get higher scores than *DJIA* index and *HRPM* is the best. That is, our strategy could gain more profit under the same risk. When it goes to *MDD* and *DDR*, the results show that *HRPM* bears the least risk, even lower than *UCRP*. The risk of our method is acceptable, although a little higher than the *DJIA* index.

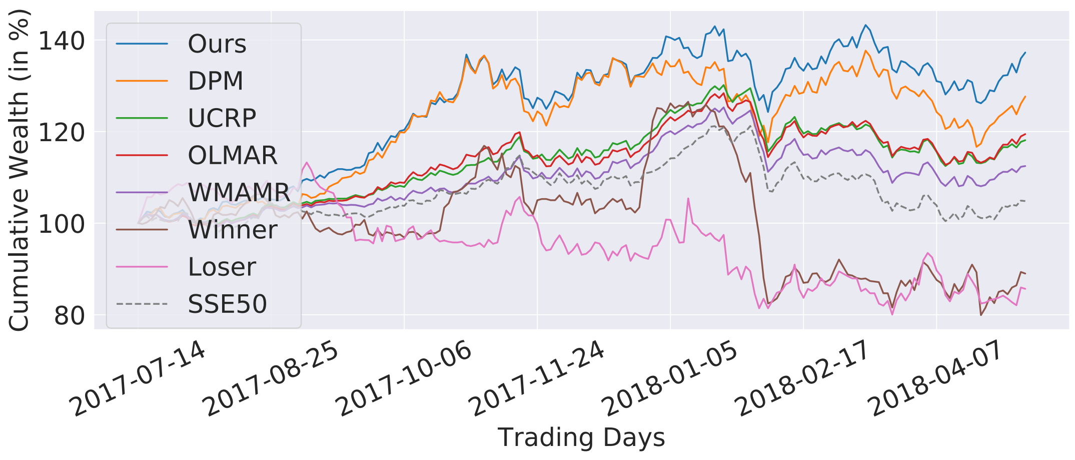


FIGURE 5.4: The portfolio value in the China market.

	ARR(%)	ASR	MDD	DDR
HRPM, $\eta = 0$	17.092	1.207	0.191	1.631
HRPM, $\eta = 0.01$	25.172	1.201	0.183	2.275
HRPM, $\eta = 0.05$	27.089	1.246	0.117	2.403
HRPM, $\eta = 0.1$	20.33	1.346	0.124	1.754
DJIA	*20.823*	*1.206*	*0.116*	*2.547*

TABLE 5.4: Ablation on the effect of entropy in the U.S. market

Backtest Results in the China Market. Figure 5.4 shows cumulative wealth vs. trading days in the China market. As we can see, *SSE50* index first rises slowly, but from then on it falls back to the starting point. During the rising stage, *HRPM* keeps the best performance and the longest time. When the portfolio values of other methods decline, our strategy still hovers at the peak. This phenomenon demonstrates that *HRPM* is not only superior to all the baselines, it is also robust under different market conditions relatively.

Table 5.3 compares all metrics of different strategies in detail. On all the four metrics, our *HRPM* achieves the best among all the strategies. Specifically, *HRPM* outperforms *DPM* by 13% and outperforms market by 42% on *ARR*. Even considering return with risk by the *ASR* metric, our strategies work better than others. Different from the performance in the U.S. market, *HRPM* not only reaches the least risk among these strategies but also performs better than the market on *MDD* and *DDR*. It means that our strategy is able to adapt to volatile markets with low risk.

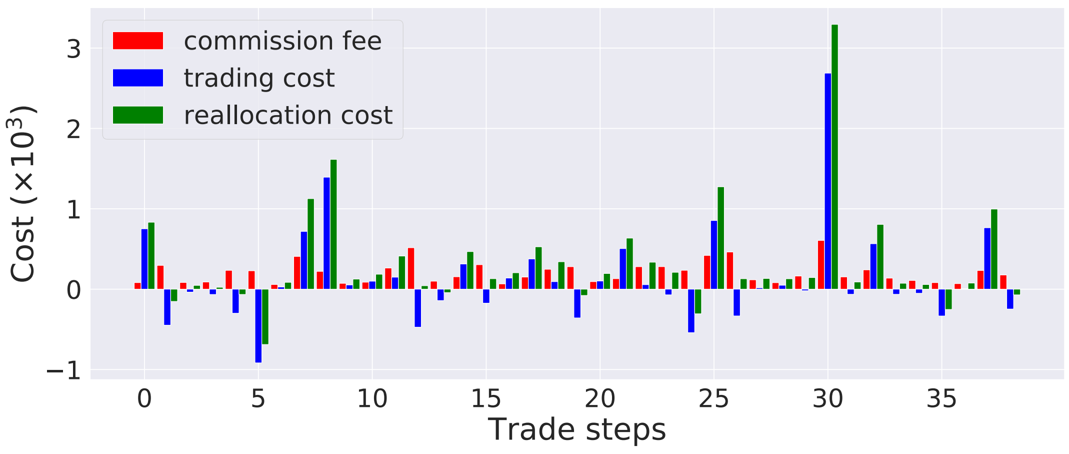


FIGURE 5.5: Trading cost during the trading periods.

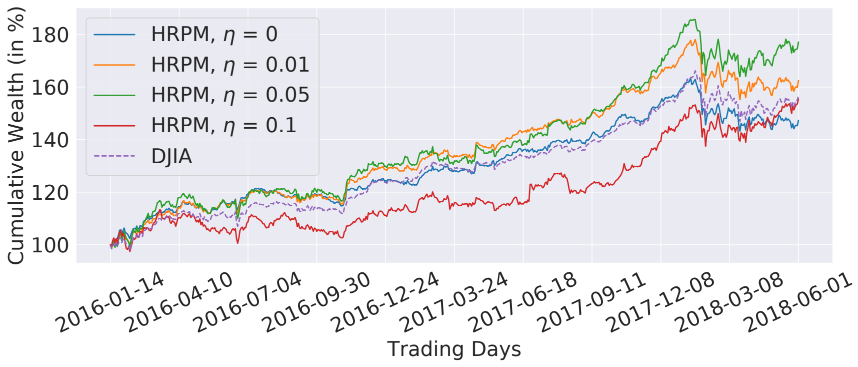
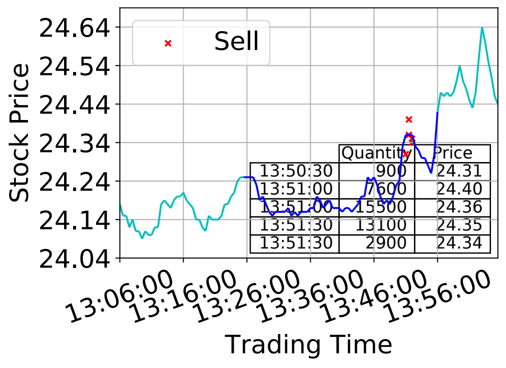


FIGURE 5.6: The effect of the entropy bonus in the U.S. Market.

5.5.4 Ablation Study

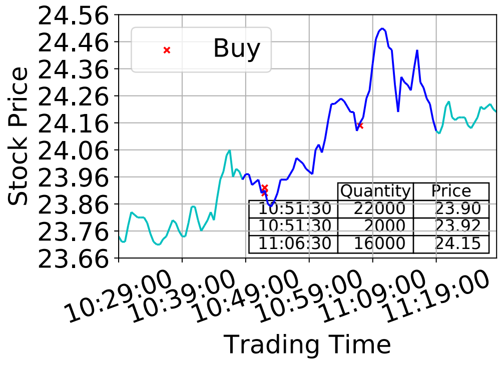
In our ablation study, we mainly answer two questions: (1) why the commission fee is not enough to account for the actual trading cost? (2) Does the entropy bonus work?

Commission Fee is Not Enough. In Fig. 5.5, we show the trading cost at each trading period of testing in the China market. The x-axis is the index of each trading period in testing, and the y-axis is the cost. Note that a negative trading cost means gaining profit in the trading period. A negative trading cost happens when the submitted orders are finished at a better price than the target price. We can see that the commission fee is only a small part of the total trading cost in most transactions, especially in large transactions.



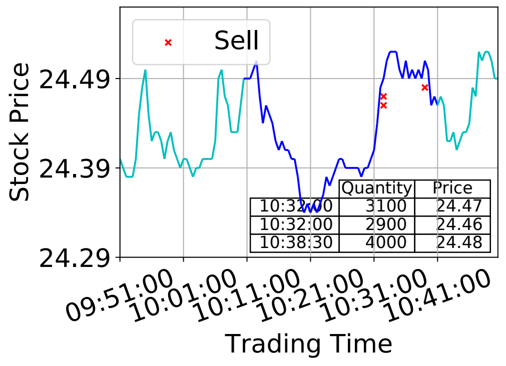
	Quantity	Price
13:50:30	900	24.31
13:51:00	7600	24.40
13:51:00	15500	24.36
13:51:30	13100	24.35
13:51:30	2900	24.34

(A) Selling 40,000 shares



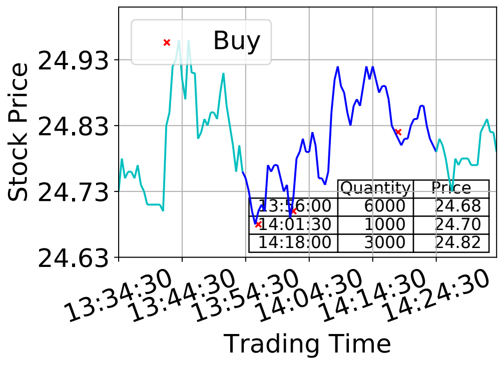
	Quantity	Price
10:51:30	22000	23.90
10:51:30	2000	23.92
11:06:30	16000	24.15

(B) Buying 40,000 shares



	Quantity	Price
10:32:00	3100	24.47
10:32:00	2900	24.46
10:38:30	4000	24.48

(C) Selling 10,000 shares



	Quantity	Price
13:56:00	6000	24.68
14:01:30	1000	24.70
14:18:00	3000	24.82

(D) Buying 10,000 shares

FIGURE 5.7: The interpretation of the low-level policy. The line is the mid-price of the market. The dark blue part is the trading window, and the light blue part shows the price before and after. The table inside the figure shows the finished trades.

Entropy Bonus. We show the effect of the entropy bonus in Fig. 5.6. As η in the entropy bonus term in reward gets higher, the agent would tend to give portfolios with more diversity. While η is small, the agent might be more likely to hold only a few stocks. Consequently, we may get a balance between the risk-control and profit by controlling η. As Table 5.4 shows, *HRPM* with $\eta = 0.05$ reaches the maximum ARR. Note that *HRPM* with $\eta = 0.1$ has the lowest MDD than others. By increasing the effect of the entropy, we can gain more profit relative to the risk.

5.5.5 Trading Strategy Interpretation

Here, we try to interpret the underlying investment strategies of the low-level policy. We show the trades that happened in the trading period in Fig. 5.7. Fig. 5.7(a)(c) shows that the low-level policy tries to sell orders at a relatively high position. It first sells part of the quantity at the beginning of the trading period. Then it sells the rest at the following local maximum. In this way, the low-level policy can avoid missing potential growth and decline. We can get the same conclusion for buying in Fig. 5.7(b)(d).

5.6 Chapter Summary

In this chapter, we focus on the problem of portfolio management with trading cost via deep reinforcement learning. We propose a hierarchical reinforced stock trading system (*HRPM*). Concretely, we build a hierarchy of portfolio management over trade execution and train the corresponding policies. The high-level policy gives portfolio weights and invokes the low-level policy to sell or buy the corresponding shares within a short time window. Extensive experimental results in the U.S. market and China market demonstrate that *HRPM* achieves significant improvement against many state-of-the-art approaches.

In Part II, we mainly explore the collaboration among agents, especially the collaboration between the high-level policy and the low-level policy. In the following part, we reciprocation amongst agents, especially motivated by the teacher-student framework, i.e., the curriculum learning in MARL.

Part III

Reciprocation amongst agents

Chapter 6

Towards Skilled Population Curriculum for Multi-agent Reinforcement Learning[1]

6.1 Introduction

In Part III of this work, we focus on the concept of reciprocity among agents in multi-agent reinforcement learning (MARL) and its application in curriculum learning. Reciprocity refers to the mutual exchange of knowledge or skills between agents, with each agent playing the role of both a teacher and a student. This approach has been inspired by the teacher-student framework, which has been successfully applied to single-agent reinforcement learning to accelerate the learning process.

In the context of MARL, reciprocity can be seen as a way to leverage the heterogeneity among agents to facilitate knowledge transfer and accelerate learning. By allowing agents to teach and learn from each other, we can exploit their complementary strengths and compensate for their weaknesses. This can lead to a more robust and efficient learning process and enable the agents to achieve higher levels of performance.

[1]The work in this chapter has been published in [82]

One of the most promising applications of reciprocity in MARL is curriculum learning. Curriculum learning is a training strategy in which agents are presented with a sequence of tasks of increasing difficulty, with each task building on the knowledge and skills acquired in the previous tasks. This approach has been shown to improve the learning efficiency and generalization ability of single-agent RL algorithms. In the context of MARL, curriculum learning can be extended to exploit the heterogeneity among agents and facilitate the transfer of knowledge and skills between them.

However, learning effective policies with sparse reward from scratch for large-scale multi-agent systems remains challenging. One of the challenges is the exponential growth of the joint observation-action space with an increasing number of agents. In addition, sparse reward signal requires a large number of training trajectories, posing difficulties in applying existing MARL algorithms directly to complex environments. As a result, these algorithms may produce agents that do not collaborate with each other, even when it would be of significant benefit [83, 84].

There are several lines of research related to the large-scale MARL problem with sparse reward, including reward shaping [85], curriculum learning [86], and learning from demonstrations [87]. Among these approaches, the curriculum learning paradigm, in which the difficulty of experienced tasks and the population of training agents progressively grow, shows particular promise. In *automatic* curriculum learning (ACL), a teacher (curriculum generator) learns to adjust the complexity and sequencing of tasks faced by a student (curriculum learner). Several works have even proposed *multi-agent* ACL algorithms, based on approximate or heuristic approaches to teaching, such as DyMA-CL [88], EPC [89], and VACL [86]. However, these approaches rely on a framework of an off-policy student with replay buffer that ignores the forgetting problem that arises when the agent population size grows, or make a strong assumption that the value of the learned policy does not change when agents switch to a different task. Moreover, the teacher in these approaches still faces a non-stationarity problem due to the ever-changing student strategies. Another class of larger-scale MARL solutions is hierarchical learning, which utilizes temporal abstraction to decompose a task into a hierarchy of subtasks. This includes skill discovery [90], option as response [91], role-based MARL [92], and two levels of abstraction [93]. However, these approaches mostly focus on one specific task with a fixed number of agents and do not consider the transferability of learned skills. In this paper, we provide our insight into this question:

Whether an elaborate combination of principles from ACL and hierarchical learning can enable ***complex*** *cooperation with* ***sparse reward*** *in* ***MARL****?*

Specifically, we present a novel automatic curriculum learning algorithm, Skilled Population Curriculum (SPC), that addresses the challenges of learning effective policies for large-scale multi-agent systems with sparse reward. The core idea behind SPC is to encourage the student to learn skills from tasks with different numbers of agents, akin to how team sports players train by gradually increasing the difficulty of tasks and the number of coordinating players. To achieve this, SPC is implemented with three key components:

The contextual bandit teacher uses an RNN-based [94] imitation model to represent student policies and generate the bandit's context.

Population-invariant communication in the student module is implemented to handle varying number of agents across tasks. By treating each agent's message as a word and using a self-attention communication channel [95], SPC supports an arbitrary number of agents to share messages.

A hierarchical skill framework is used in the student module to learn transferable skills in the sparse reward setting, where agents communicate on the high-level about a set of shared low-level policies.

Empirical results show that our method achieves state-of-the-art performance in several tasks in Multi-agent Particle Environment (MPE) [36] and the challenging 5vs5 competition in Google Research Football (GRF) [96]. [2]

6.2 Preliminaries

Multi-armed Bandit. Multi-armed bandits (MABs) are a simple but very powerful framework that repeatedly makes decisions under uncertainty. In this framework, a learner performs a sequence of actions and immediately observes the corresponding reward after each action. The goal is to maximize the total reward

[2]The source code and a video demonstration can be found at `https://sites.google.com/view/marl-spc/`.

over a given set of K actions and a specific time horizon T. The measure of success in MABs is often determined by the regret, which is the difference between the cumulative reward of an MAB algorithm and the best-arm benchmark. One well-known MAB algorithm is the Exp3 algorithm [97], which aims to increase the probability of selecting good arms and achieves a regret of $O(\sqrt{KT\log(K)})$ under a time-varying reward distribution. Another related concept is the contextual bandit problem [98], where the learner makes decisions based on prior information as the context.

6.3 Skilled Population Curriculum

In this section, we first provide a formal definition of the curriculum-enhanced Dec-POMDP framework, which formulates the MARL with curriculum problem under the Dec-POMDP framework. We then present our multi-agent ACL algorithm, Skilled Population Curriculum (SPC), as shown in Fig. 6.1. In the following subsections, we establish the curriculum learning framework in Sec. 6.3.1, and then present a contextual multi-armed bandit algorithm as the teacher to address the non-stationarity in Sec. 6.3.2. Lastly, we introduce the student with transferable skills and population-invariant communication to tackle the varying number of agents and the sparse reward problem in Sec. 6.3.3.

6.3.1 Problem Formulation

We consider environments from multi-agent automatic curriculum learning problems are equipped with parameterized task spaces and thus can be modeled as curriculum-enhanced Dec-POMDPs.

Definition 6.1 (Curriculum-enhanced Dec-POMDP)**.** A curriculum-enhanced Dec-POMDP is defined by a tuple $\langle\Phi,\mathcal{M}\rangle$, where Φ and $\mathcal{M}$ represent a task space and a Dec-POMDP, respectively. Given the task ϕ, the Dec-POMDP $\mathcal{M}(\phi)$ is presented as $\left\{n^\phi,\boldsymbol{S}^\phi,\boldsymbol{A}^\phi,P^\phi,r^\phi,O^\phi,\boldsymbol{\Omega}^\phi,\gamma^\phi\right\}$. The superscript ϕ denotes that the Dec-POMDP elements are determined by the task ϕ. Note that task ϕ can be a few parameters of the environment or task IDs in a finite task space. *In a curriculum-enhanced Dec-POMDP, the objective is to improve the student's performance on the target tasks through the sequence of training tasks given by the teacher..*

Let τ denote a trajectory whose unconditional distribution $\mathrm{Pr}_{\mu}^{\pi,\phi}(\tau)$ (under a policy π and a task ϕ with initial state distribution $\mu(s_0)$) is $\mathrm{Pr}_{\mu}^{\pi,\phi}(\tau) = \mu(s_0) \sum_{t=0}^{\infty} \pi(a_t \mid s_t) P^{\phi}(s_{t+1} \mid s_t, a_t)$. We use $p(\phi)$ to represent the distribution of target tasks and $q(\phi)$ to represent the distribution of training tasks at each task sampling step. We consider the joint agents' policies $\pi_\theta(a|s)$ and $q_\psi(\phi)$ parameterized by θ and ψ, respectively. The overall objective to maximize in a curriculum-enhanced Dec-POMDP is:

$$\begin{aligned} J(\theta, \psi) &= \mathbb{E}_{\phi \sim p(\phi), \tau \sim \mathrm{Pr}_{\mu}^{\pi}} \left[R^{\phi}(\tau) \right] \\ &= \mathbb{E}_{\phi \sim q_\psi(\phi)} \left[\frac{p(\phi)}{q_\psi(\phi)} V(\phi, \pi_\theta) \right] \end{aligned} \tag{6.1}$$

where $R^{\phi}(\tau) = \sum_t \gamma^t r^{\phi}(s_t, a_t; s_0)$ and $V(\phi, \pi_\theta)$ represents the value function of π_θ in Dec-POMDP $\mathcal{M}(\phi)$. However, when optimizing $q_\psi(\phi)$, we cannot get the partial derivative $\nabla_\psi J(\theta, \psi) = \nabla_\psi \sum_\tau \frac{1}{q_\psi(\phi)} R^{\phi}(\tau) \mathrm{Pr}_{\mu}^{\pi,\phi}(\tau)$[3] since the reward function and the transition probability function w.r.t number of agents are non-parametric, non-differentiable, and discontinuous in most MARL scenarios.

Thus, we use the non-differentiable method, i.e., multi-armed bandit algorithms, to optimize $q_\psi(\phi)$, and use an RL algorithm (the student) in alternating periods to optimize $\pi_\theta(a|s)$. However, there are three key challenges in solving this problem: (1) The teacher is facing a non-stationarity problem due to the ever-changing student's strategies. (2) The student will forget the old tasks and need to re-learn them. Some tasks can be the prerequisites of other tasks, while some can be inter-independent and parallel. (3) There is a lack of a general student framework to deal with the varying number of agents across tasks and the sparse reward problem.

6.3.2 Teacher as a Non-Stationary Contextual Bandit

As previously discussed, the teacher faces a non-stationarity problem due to the ever-changing student's strategies during the learning process. Specifically, as the student learns across different tasks in different learning stages, the teacher will observe varying student performance when providing the same task, resulting in a time-varying reward distribution for the teacher. In addition, the student may forget previously learned policies. To mitigate this problem, the teacher should

[3] $p(\phi)$ is not in the partial derivative since it is a fixed distribution.

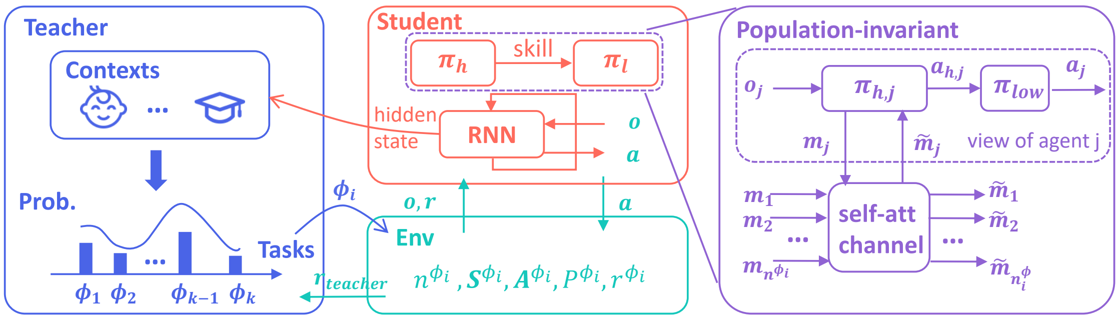


FIGURE 6.1: The overall framework of SPC. It consists of three parts: configurable environments, a teacher, and a student. Left. The teacher is modeled as a contextual multi-armed bandit. At each teacher timestep, the teacher chooses a training task from the distribution of bandit actions. Mid. The student is endowed with a hierarchical skill framework and population-invariant communication. It is trained with MARL algorithms on the training tasks. The student returns not only the hidden state of its RNN imitation model as contexts to the teacher, but also the average discounted cumulative rewards on the testing task. Right. The student learns hierarchical policies, with the population-invariant communication taking place at the high-level, implemented with a self-attention communication channel to handle the messages from a varying number of agents. The agents in the student share the same low-level policy.

balance the exploitation of tasks that have been found to benefit the student's performance on the target tasks, with the exploration of tasks that may not directly facilitate the student's learning.

Fortunately, we notice that the non-stationarity stems from the student, which can be mitigated with a contextual bandit which embeds the student policy into the context. As shown in Fig. 6.1 Left, the teacher utilizes the student's policy representation as the context and chooses a task from the distribution of training tasks. Specifically, we extend the Exp3 algorithm [97] by incorporating contexts through a two-step online clustering process [99]. The context, represented by x, is the student's policy representation. The teacher's action is a specific task, denoted by ϕ, and the teacher's reward is the return of the student in the target tasks. The teacher's algorithm is outlined in Alg. 2. During the sampling stage (steps 1-5), the teacher selects a task for the student's training. In the training stage (steps 6-7), the teacher adjusts the parameters based on the evaluation reward received from the student.

Algorithm 2: Teacher Sampling and Training

```
Input: Context $x$, the number of Clusters $N_c$, $N_c$ instances of Exp3 with task
  distribution $w(\phi_k, c)$ for $k = 1, \dots, K$ and for $c = 1, \dots, N_c$, learning rate $\alpha$,
  a buffer maintaining the historical contexts
Output: $\mathcal{M}(\phi) = \left\{n^\phi, \boldsymbol{S}^\phi, \boldsymbol{A}^\phi, P^\phi, r^\phi, O^\phi, \boldsymbol{\Omega}^\phi, \gamma^\phi\right\}$, the teacher bandit
  parameters
Sampling
1. Get the the context $x$, and save it to the buffer
2. Run the online cluster algorithm and get the index of the cluster center $c(x)$
3. Let the active Exp3 instance be the instance with index $c(x)$
4. Set the probability $p(\phi_k, c(x)) = \frac{(1-\alpha) w(\phi_k, c(x))}{\sum_{j=1}^{K} w(\phi_k, c(x))} + \frac{\alpha}{K}$ for each task $\phi_k$
5. Sample a new task according to the distribution of $p_{\phi_k, c}$
Training
6. Get the return (discounted cumulative rewards) from student testing $r$
7. Update the active Exp3 instance by setting $w(\phi_k, c(x)) = w(\phi_k, c(x)) e^{\alpha r / K}$
```

6.3.2.1 Context Representation

Upon analysis, it is essential to learn an effective representation for the student's policy as the context. One straightforward representation is to use the student parameters θ directly as the context. However, the number of parameters is too large to be used as the input of neural network if we change the student's architecture. Therefore, we propose an alternative method.

A principle for learning a good representation of a policy is *predictive representation*, which means the representation should be accurate to predict policy actions given states. In accordance with this principle, we utilize an imitation function through supervised learning. Supervised learning does not require direct access to reward signals, making it an attractive approach for reward-agnostic representation learning. Intuitively, the imitation function attempts to mimic low-level policy based on historical behaviors. In practice, we use an RNN-based imitation function $f_{im} : \mathcal{S} \times \mathcal{A} \rightarrow [0, 1]$. Since recurrent neural networks are theoretically Turing complete [100], their internal states can be used as the representation of the student's policy. We train this imitation function by using the negative cross entropy objective $\mathbb{E}[\log f_{im}(s, a)]$.

6.3.2.2 Regret Analysis

In this subsection, we demonstrate that the proposed teacher algorithm has a regret bound of $\mathbb{E}[R(T)] = O\left(T^{2/3}(LK\log T)^{1/3}\right)$, where T is the number of total rounds, L is the Lipschitz constant, and K is the number of arms (the number of the teacher's actions). The regret analysis is used to justify the usage of the bandit algorithm in the non-stationary setting. The regret bound represents the optimality of SPC, as the teacher's reward is the return of the student in the target tasks.

First, we introduce the Lipschitz assumption about the generalization ability of the task space.

Assumption 6.1 (Lipschitz continuity w.r.t the context)**.** Without loss of generality, the contexts are mapped into the $[0,1]$ interval, so that the expected rewards for the teacher are Lipschitz with respect to the context.

$$\begin{gathered}|r(\phi \mid x) - r(\phi \mid x')| \leq L \cdot |x - x'| \\ \text{for any arm } \phi \in \Phi \text{ and any pair of contexts } x, x' \in \mathcal{X}\end{gathered} \tag{6.2}$$

where L is the Lipschitz constant, and $\mathcal{X}$ is the context space.

This assumption suggests that for any policy trained on a set of tasks, the rate at which performance improves is not faster than the rate at which the policy changes. This is a realistic assumption, as we cannot expect the student to achieve a significant improvement on a task with only a few training steps under a new context. We use an existing contextual bandit algorithm for a limited number of contexts [97] (see Appendix B.1) and Lemma 6.1 as a foundation for proving Theorem 6.3.2.2.

Lemma 6.1. *Alg. 4 has a regret bound of* $\mathbb{E}[R(T)] = \mathcal{O}(\sqrt{TK|\mathcal{X}|\log K})$.

Lemma 6.1 introduces a square root dependence on $|\mathcal{X}|$ if separate copies of Exp3 are run for each context [97]. This motivates us to address the large context space by utilizing discretization techniques.

theoremregretbound Consider the Lipschitz contextual bandit problem with contexts in $[0,1]$. The Alg. 2 yields regret $\mathbb{E}[R(T)] = O\left(T^{2/3}(LK\ln T)^{1/3}\right)$.

Proof. See Appendix B.2. □

In practice, the high-dimensional context space cannot be discretized using a uniform mesh in $[0, 1]$ as in the proof of Theorem 6.3.2.2. To address this issue, we utilize the Balanced Iterative Reducing and Clustering using Hierarchies (BIRCH) online clustering algorithm [99] to discretize the context space. BIRCH is an efficient and easy-to-update algorithm that can effectively cluster large datasets. In this case, it is used to cluster the high-dimensional RNN-based policy representation. The resulting clusters can be seen as an approximation of a uniform mesh.

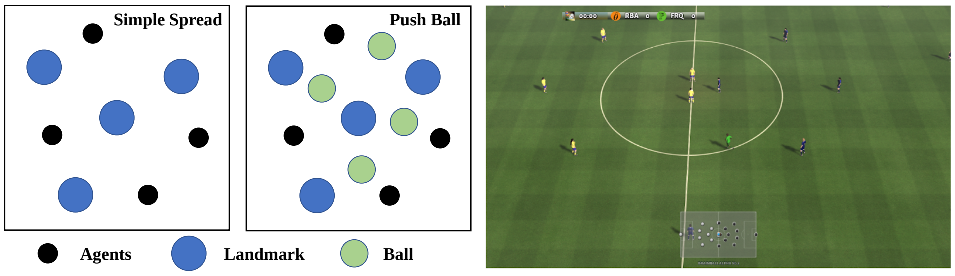
6.3.3 Student with Population-Invariant Skills

We propose a population-invariant skill framework to address the challenges of varying number of agents and sparse reward problem. This framework allows agents to communicate via a self-attention channel, enabling them to learn transferable skills across different tasks. The student module is designed to be algorithm-agnostic and is orthogonal to any state-of-the-art MARL algorithm. While there have been some efforts in the literature to address the varying number of agents [101, 102], these approaches heavily rely on prior knowledge of the environments.

Population-Invariant Teamwork Communication. In order to enable the population-invariant property and learn tactics among agents, we introduce communication. Leveraging the transformer architecture's capability to process inputs of varying lengths [95], we incorporate self-attention into our communication mechanism. As illustrated in Fig. 6.1 Right, each agent j receives an observation o_j and encodes it into a message vector $m_j = f(o_j)$ which is then sent through a self-attention channel, where f is an observation encoder function.

The channel aggregates all messages and sends the new message vector, $\tilde{m}_j$, through the self-attention mechanism. Concretely, given the channel input $\mathbf{M} = [m_1, m_2, \cdots, m_n] \in R^{n\times d_m}$, and the trainable weight of the channel $\mathbf{W}_Q, \mathbf{W}_K, \mathbf{W}_V \in R^{d_m \times d_m}$, we obtain three distinct representations: $\mathbf{Q} = \mathbf{M}\mathbf{W}_Q, \mathbf{K} = \mathbf{M}\mathbf{W}_K, \mathbf{V} = \mathbf{M}\mathbf{W}_V$. Then the output messages are

$$\tilde{\mathbf{M}} = \text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_m}}\right)\mathbf{V} \tag{6.3}$$



(A) (B)

FIGURE 6.2: (a) Multi-agent Particle Environment. (b) Google Research Football.

where d_m is the dimension of the messages. As the dimensions of the trainable weight are independent of the number of agents, our student models can leverage the population-invariant property to effectively learn tactics.

Transferable Hierarchical Skills. As depicted in the dotted box in Fig. 6.1 Right, after receiving the new messages $\tilde{m}_j$ from the channel, each agent employs a high-level action (skill) $a_{h,j} = \pi_{h,j}(o_j, \tilde{m}_j)$ to execute the low-level policy $a_j = \pi_{low}(o_j, a_{h,j})$. In this work, we generalize the high-level action (skill) $a_{h,j}$ to a continuous embedding space, so that the skill can be either a latent continuous vector as in DIAYN [47], or a categorical distribution for sampling discrete options [49].

Implementation. We implement the high- and low-level policies in the student with Proximal Policy Optimization (PPO) [12]. Following the common practice proposed in [103], the high-level policy for each agent is learned independently, whereas the low-level policies share parameters, as the fundamental action pattern should be consistent among different agents. The low-level agents are rewarded by the environment, while the high-level policy is trained to take actions at fixed intervals. Within this interval, the cumulative low-level reward is used as the high-level reward. When using a categorical distribution to enable discrete skills, we sample an “option” from the distribution and provide the corresponding one-hot embedding to the low-level policy.

6.4 Further Related Work

Automatic Curriculum Learning in MARL. Curriculum learning is a training strategy that mimics the human learning process by organizing tasks based on their difficulty level [104]. The selection of tasks is formulated as a Curriculum Markov Decision Process (CMDP) [105]. Automatic Curriculum Learning mechanisms aim to learn a task selection function based on past interactions, such as ADR [106, 107], ALP-GMM [108], SPCL [109], GoalGAN [110], PLR [111, 112], SPDL [113], CURROT [114], and graph-curriculum [115]. Recently, several MARL curriculum learning frameworks have been proposed, such as open-ended evolution [116–118], population-based training [5, 119], meta-learning [120, 121] and training with emergent curriculum [4, 104, 122, 123]. In summary, these frameworks share a common principle of an automatic curriculum that continually generates improved agents through selection pressure among a population of self-optimizing agents.

Hierarchical MARL and Communication. Hierarchical reinforcement learning (HRL) has been extensively studied to address the issue of sparse reward and facilitate transfer learning. Single-agent HRL focuses on learning the temporal decomposition of tasks, either by learning subgoals [42, 44, 124–126] or by discovering reusable skills [127–130]. Recent developments in hierarchical MARL have been discussed in Sec. 6.1. In multi-agent settings, communication has been effective in promoting cooperation among agents [15–17, 19, 21, 22, 25]. However, current approaches that extend HRL to multi-agent systems or utilize communication are limited to a fixed number of agents and lack the ability to transfer to different agent counts.

Multi-task RL. Multi-task reinforcement learning (MTRL) is a promising approach to train effective real-world agents. Some work focus on language-conditioned RL, where natural language phrases have been used as part of task descriptions in the context of several single task RL setups like goal-oriented RL [131], grounded language acquisition [132, 133] , and instruction following [134]. Several works have also focused on learning compositional models for multi-task learning [135, 136].

6.5 Experiments

To demonstrate the effectiveness of our approach, we conduct experiments on several tasks in two environments: Simple-Spread and Push-Ball in the Multi-agent Particle Environment (MPE) [36], and the challenging 5vs5 task of the Google

Research Football (GRF) environment [96]. We aim to investigate the following research questions:

Q1: *Is curriculum learning necessary in complex large-scale MARL problems?* (Sec. 6.5.2)

Q2: *Can SPC outperform previous curriculum-based MARL methods? If so, which components of SPC contribute the most to performance gains?* (Sec. 6.5.3)

Q3: *Can SPC effectively learn a curriculum for the student?* (Sec. 6.5.4)

6.5.1 Environments, Baselines and Metric

Environments. In the GRF 5vs5 scenario, we control four agents, excluding the goalkeeper, to compete against the built-in AI opponents. Each agent observes a compact encoding, consisting of a 115-dimensional vector that summarizes various aspects of the game, such as player coordinates, ball possession and direction, active players, and game mode. The available action set for an individual agent includes 19 discrete actions, such as idle, move, pass, shoot, dribble, etc. The GRF provides two types of rewards: scoring and checkpoints, to encourage agents to move the ball forward and make successful shots. Additionally, we include a shooting reward in the challenging GRF 5vs5 task. For curriculum, we select several basic scenarios in GRF, including 3vs3, Pass-Shoot, 3vs1, and Empty-Goal.

In MPE, we investigate Simple-Spread and Push-Ball (see Fig. 6.2a). In Simple-Spread, there are n agents that need to cover all n landmarks. Agents are penalized for collisions and only receive a positive reward when all the landmarks are covered. In Push-Ball, there are n agents, n balls, and n landmarks. The agents must push the balls to cover each landmark. A success reward is given after all the landmarks have been covered.

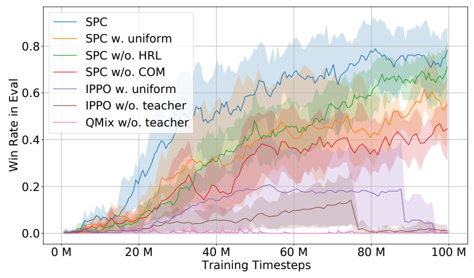
Baselines. We compare our approach to the following methods in Table 6.1 as baselines[4]:

Metric. To evaluate the performance of our approach in the GRF 5vs5 scenario, we use metrics beyond just the mean episode reward, as this alone may not accurately

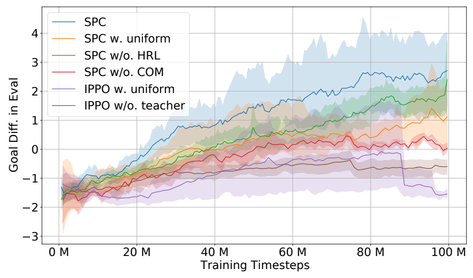
[4]We also run CDS [137] and CMARL [138], but we have not included their performance because the goal difference reported in CMARL [138] is relatively low compared to our method.

TABLE 6.1: Baseline algorithms.

Categories	**Methods**
MARL	QMIX [41]
	IPPO [139]
Curriculum-based	IPPO with uniform task sampling
	VACL [86]
Ablation Study	SPC with uniform task sampling
	SPC without HRL and COM



(A) Win Rate



(B) Goal Difference

FIGURE 6.3: The evaluation performance of various methods on 5vs5 football competition.

reflect the agents' performance. Specifically, we use the win rate and the average goal difference, which is calculated as the number of goals scored by the MARL agents minus the number of goals scored by the opposing team.

We evaluate the performance of MARL algorithms to justify the need for curriculum learning in complex large-scale MARL problems. To ensure a fair comparison, we modify VACL by removing the centralized critic for MPE tasks. In all experiments, we use individual Proximal Policy Optimization (IPPO) as the backend MARL algorithm. To ensure the robustness of our results, we conduct experiments on a 30-node cluster, with one node containing a 128-core CPU and four A100 GPUs. Each trial of the experiment is repeated over five seeds and runs for 1-2 days.

6.5.2 The Necessity of Curriculum Learning

Our experiments first show that in simple environments, such as MPE, students can directly learn to complete the task without the need for curriculum. For MPE

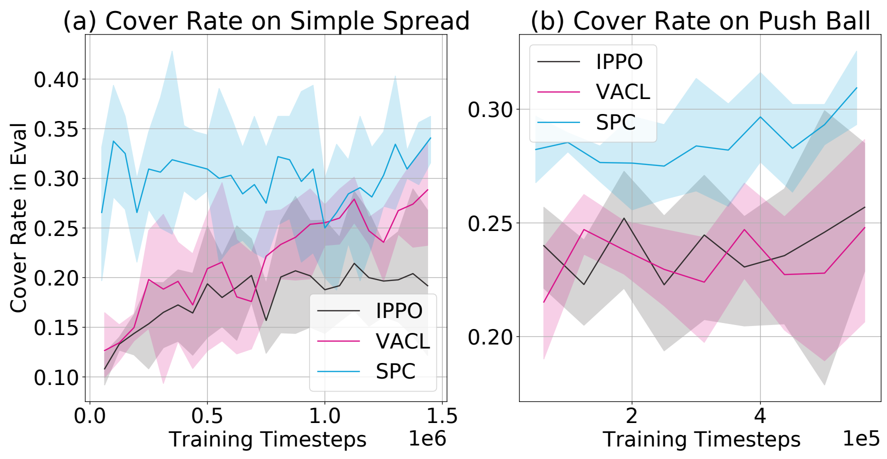


FIGURE 6.4: The evaluation performance of various methods on MPE.

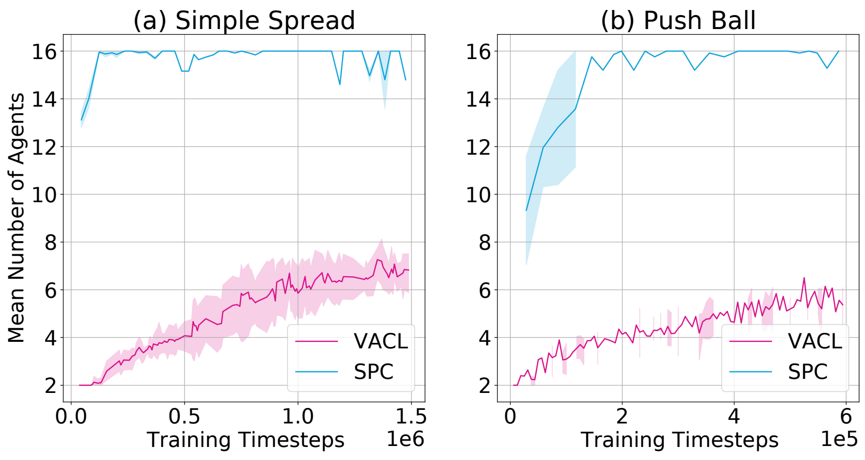
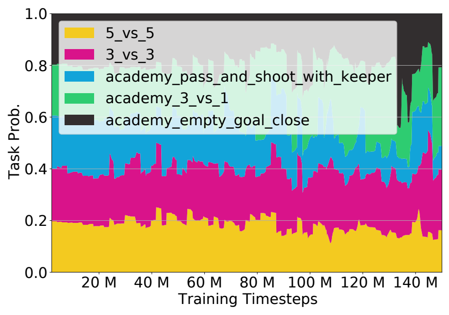
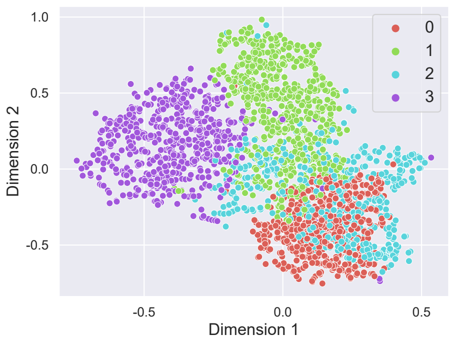


FIGURE 6.5: The changes in the number of agents on MPE.



(A) The task distribution of SPC during training.



(B) The visualization of contexts.

FIGURE 6.6: Visualization of Learned Curriculum.

experiments, we randomly select a starting state and the episode ends after a fixed number of maximum steps. Specifically, the task space consists of n agents, where $n \in \{2, 4, 8, 16\}$, and the maximum allowed steps is set to 25. All evaluations are performed on the target task, with $n = 16$. IPPO is trained and evaluated directly on the target task, and results in Fig. 6.4 demonstrate that it performs similarly to the VACL algorithm. Additionally, we observe that the SPC approach only achieves a slightly higher coverage rate than the baseline methods. Furthermore, we investigate the probability variation of different population sizes, shown in Fig. 6.5. We observe that the curriculum provided by SPC is approaching the target task. These results suggest that in simple environments where the student can learn to directly complete the task, curriculum learning may not be necessary.

However, when it comes to more complex scenarios, such as the 5vs5 task in GRF, our results demonstrate that curriculum learning is a promising solution. As shown in Fig. 6.3a, without curriculum learning, QMix and IPPO cannot perform well in the 5vs5 scenario, and IPPO is slightly better than QMix. In Fig. 6.3b, we omit the curve of QMix as its mean score is low and affects the presentation of the figure. The reason could be that QMix is an off-policy MARL algorithm, which would rely heavily on the replay buffer. However, in such sparse reward scenarios, the replay buffer has much less effective samples for QMix to learn. For example, the replay buffer would contain tons of zero-score samples, leading to a non-promising performance. Meanwhile, IPPO, with its on-policy nature, is able to achieve better sample efficiency and outperform off-policy algorithms like QMix in such scenarios. Though MARL methods can achieve good performance in basic scenarios in GRF, they fail to solve complex scenarios such as the 5vs5 task. Therefore, curriculum learning is a promising solution to the complex large-scale MARL problem.

6.5.3 Performance and Ablation Study

Our study demonstrates that SPC outperforms VACL in MPE tasks. Instead of training with a continuous relaxation of the population size variable as in VACL, our bandit teacher achieves a higher success rate at test time, since the population size is a discrete variable in nature. Furthermore, the curriculum provided by SPC is effective in exploring the task space and converge to the target task when the task is relatively simple and curriculum is not necessary, as shown in Fig. 6.5.

In GRF experiments, we do not include VACL in our baselines in the GRF, as its implementation relies heavily on prior knowledge of specific scenarios, such as the thresholds to divide the learning process. Fig. 6.6 indicates that SPC has higher win rate and goal difference than IPPO with uniform task sampling in the 5vs5 competition. These experiments demonstrate that when the teacher is rewarded by the student's performance, a bandit-based teacher can exploit the student's learning stage and provide suitable training tasks.

In our ablation study, we examine the impact of two key components of our SPC algorithm: the contextual multi-armed bandit teacher and the hierarchical structure of the student framework. By replacing the former with uniform task sampling and removing the latter, As shown in Fig. 6.3a and Fig. 6.3b, SPC can achieve a higher win rate and a greater score difference than SPC with uniform and SPC without HRL. Furthermore, SPC with uniform task sampling outperforms IPPO with uniform task sampling. This highlights the importance of HRL in the 5vs5 football competition, and suggests that both the contextual multi-armed bandit and the hierarchical structure contribute equally to the performance of SPC. When removing HRL and bandit, the performance degradation w.r.t. SPC are similar. However, it should be noted that SPC with uniform task sampling has a larger variance in performance than SPC without HRL, indicating that uniform sampling may introduce more undesired tasks for student training. Overall, these results further justify the necessity of SPC in complex large-scale MARL problems[5].

6.5.4 Visualization of Learned Curriculum

We visualize the distribution of task sampling of SPC during training based on a selected trial as shown in Fig. 6.6a. At the beginning of training, the task probability appears to be near-uniform, as the teacher explores the task space and keeps track of the student's learning status, acting as an anti-forgetting mechanism. As training progresses, the probabilities change over time. For example, the proportions of 3vs1 and Empty-Goal tasks gradually drop as the student becomes proficient in these scenarios. We also visualize the distribution of contexts in Fig. 6.6b using t-SNE [140], where the contexts are collected and stored in a buffer. We divide the contexts into four classes according to the index, and different parts represent different contexts of the final student policy representation.

[5]We also demonstrate the performance of SPC in the GRF 11vs11 full game (see Appendix B.3).

Chapter 7

Conclusions and Future Directions

7.1 Conclusion

This thesis has explored ways to enhance the efficiency of interaction and cooperation between agents in various multi-agent reinforcement learning settings. The research presented in this thesis has addressed several key challenges in MARL, including communication, collaboration, and reciprocation, by developing novel algorithms and techniques that can be applied in a wide range of real-world applications.

The key lessons that the field should learn from this thesis are as follows:

Effective communication is crucial for the success of multi-agent systems, especially in limited bandwidth environments. The information bottleneck principle introduced in Chapter 2 provides a valuable approach for agents to learn informative communication protocols and scheduling while working within bandwidth constraints. This principle can be further explored and extended to other communication scenarios in MARL.

Addressing non-stationarity in hierarchical reinforcement learning is essential for improving the performance of agents in complex, dynamic environments. The Interactive Influence-based HRL (HRL) algorithm presented in Chapter 3 demonstrates the importance of interaction between high-level and low-level policies and provides a framework for adaptively exploring non-stationary states. This approach can serve as a foundation for future research on end-to-end HRL algorithms.

Incorporating hierarchical structures in portfolio management can lead to significant improvements in trading performance. The hierarchical reinforced stock trading system (HRPM) proposed in Chapter 4 demonstrates the benefits of building a hierarchy of portfolio management over trade execution and training corresponding policies. This approach can inspire further research on hierarchical methods in finance and other domains.

Addressing scalability and sparse reward issues in multi-agent systems is crucial for the development of more sophisticated and effective algorithms. The Skilled Population Curriculum (SPC) presented in Chapter 5 offers a novel approach for tackling these challenges by incorporating a population-invariant multi-agent communication framework and using a hierarchical scheme for agents to learn skills. This work highlights the importance of addressing these issues for the continued advancement of multi-agent algorithms.

In conclusion, this thesis has made significant contributions to the field of multi-agent reinforcement learning by addressing key challenges and providing novel algorithms and techniques that can be applied in various settings. The insights gained from this research can guide future work in MARL, ultimately leading to the development of more efficient, robust, and intelligent multi-agent systems capable of tackling complex real-world challenges.

7.2 Future Directions

In the final section of this thesis, we show some potentially interesting future directions of multi-agent cooperation.

7.2.1 Foundation Model for Decision Making

Recently, transformer-based foundation models have been shown to be a powerful tool for RL to capture global dependencies or model long-term squence. The architecture of is closely related to AlphaStar [5] and OpenAI Five [141], which use transformer-based models to embed unit features for each player. Another perspective of using transformer-based models is to model an RL problem as a sequence prediction problem that models trajectories autoregressively, such as Decision Transformer (DT) [142], Trajectory Transformer (TT) [143], GATO [144], and Multi-Game DT (MGDT) [145]. In addition, Multi-Agent transformer (MAT) [146] considers the MARL problem as a sequence prediction problem that generates the optimal action of each agent sequentially.

These transformer-based models study the generality with *trajectory sequences* in *single agent* scenarios [144, 145] or focus on *a single task* with the sequences [5, 141–143, 146]. In addition, they use causal transformer decoders, which use causal masking to discard future information. We would like to advocate an interesting setting by using a transformer model without positional encoding and exploring generality with entity sequences in multi-agent scenarios.

Another related research direction is to reuse trained RL models, which is an important subfield of RL. Some approaches directly use the trained models. For example, to avoid restarting from scratch after changes in code and environment, OpenAI Five [141] used the "surgery" approach to convert a trained model to certain larger architectures with customized weight initialization. MGDT [145] and Gato [144] follow the manner of few-shot transfer learning by using a single set of weights to play many games or handle tasks in multiple domains simultaneously. Furthermore, some approaches implicitly exploit trained models, such as behavior cloning [147, 148], distillation [149, 150], and the use of the teacher-student framework [151–153]. There is a promising direction by incorporating prompt engineering into the training of foundation models for decision making.

7.2.2 Human-AI Coordination

Human-AI coordination is an important area of research that focuses on how humans and AI systems can collaborate effectively to achieve common goals. In the context of MARL, there are several potential future research directions that can help to advance the field of human-AI coordination:

Human-AI communication: One important direction is to develop better methods for human-AI communication. Effective communication is essential for successful coordination between humans and AI systems, but current communication methods are often limited. For example, AI systems may be able to communicate only in a limited set of pre-defined messages, which can make it difficult for humans to understand their intentions. Developing more flexible and natural communication methods, such as natural language processing, can improve the ability of humans and AI systems to coordinate effectively.

Explainability: Another important direction is to improve the explainability of AI systems. Humans need to understand the decision-making processes of AI systems in order to trust and effectively collaborate with them. However, many AI systems are complex and difficult to understand, which can hinder effective coordination. Developing methods for explaining the behavior of AI systems, such as visualizing the decision-making process, can improve human-AI coordination.

Trust and transparency: Trust is essential for effective human-AI coordination. Humans need to trust that AI systems will act in their best interests and will not harm them. Developing methods for ensuring transparency and accountability of AI systems, such as providing clear explanations of their decision-making processes, can improve trust and facilitate coordination.

Mixed-initiative coordination: Finally, another important direction is to develop methods for mixed-initiative coordination between humans and AI systems. Mixed-initiative coordination allows humans and AI systems to take on different roles in the decision-making process, depending on their individual strengths and weaknesses. Developing methods for adaptive mixed-initiative coordination can improve the efficiency and effectiveness of human-AI coordination.

In summary, future research directions for human-AI coordination from the perspective of MARL include improving human-AI communication, developing methods for explainability, ensuring trust and transparency, and developing methods for mixed-initiative coordination. These directions can help to improve the collaboration between humans and AI systems and enable the development of more effective and efficient systems for solving complex problems.

List of Author's Publications[1]

Conference Proceedings

- **Rundong Wang**, Xu He, Runsheng Yu, Wei Qiu, Bo An, and Zinovi Rabinovich. "Learning efficient multi-agent communication: An information bottleneck approach." In *International Conference on Machine Learning, pp. 9908-9918. PMLR, 2020.*

- **Rundong Wang**, Hongxin Wei, Bo An, Zhouyan Feng, and Jun Yao. "Commission fee is not enough: A hierarchical reinforced framework for portfolio management." In *Proceedings of the AAAI Conference on Artificial Intelligence, vol. 35, no. 1, pp. 626-633. 2021.*

- **Rundong Wang**, Runsheng Yu, Bo An, and Zinovi Rabinovich. "I2hrl: Interactive influence-based hierarchical reinforcement learning." In *Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence, pp. 3131-3138. 2021.*

- Shuo Sun, Wanqi Xue, **Rundong Wang***, Xu He, Junlei Zhu, Jian Li, and Bo An. "DeepScalper: A Risk-Aware Reinforcement Learning Framework to Capture Fleeting Intraday Trading Opportunities." In *Proceedings of the 31st ACM International Conference on Information Knowledge Management, pp. 1858-1867. 2022.*

- Xu He, Bo An, Yanghua Li, Haikai Chen, **Rundong Wang**, Xinrun Wang, Runsheng Yu, Xin Li, and Zhirong Wang. "Learning to collaborate in multi-module recommendation via multi-agent reinforcement learning without communication." In *Proceedings of the 14th ACM Conference on Recommender Systems, pp. 210-219. 2020.*

- Wei Qiu, Xinrun Wang, Runsheng Yu, **Rundong Wang**, Xu He, Bo An, Svetlana Obraztsova, and Zinovi Rabinovich. "RMIX: Learning risk-sensitive policies for cooperative reinforcement learning agents." Advances in *Neural Information Processing Systems 34 (2021): 23049-23062.*

[1]The superscript * indicates corresponding authors

- Hang Xu, **Rundong Wang**, Lev Raizman, and Zinovi Rabinovich. "Transferable environment poisoning: Training-time attack on reinforcement learning." In *Proceedings of the 20th international conference on autonomous agents and multiagent systems, pp. 1398-1406.* 2021.
- Zhenyu Shi, Runsheng Yu, Xinrun Wang, **Rundong Wang**, Youzhi Zhang, Hanjiang Lai, and Bo An. "Learning expensive coordination: An event-based deep RL approach." In *International Conference on Learning Representations. 2020.*

Journal Articles

- Bo An, Shuo Sun, and **Rundong Wang**. "Deep Reinforcement Learning for Quantitative Trading: Challenges and Opportunities." *IEEE Intelligent Systems 37, no. 2 (2022): 23-26.*
- Zhuoyi Lin, Sheng Zang, **Rundong Wang**, Zhu Sun, J. Senthilnath, Chi Xu, and Chee Keong Kwoh. "Attention over self-attention: Intention-aware re-ranking with dynamic transformer encoders for recommendation." *IEEE Transactions on Knowledge and Data Engineering (2022).*
- Yiming Li, **Rundong Wang**, Yulin Hu, Junan Yang, and Xiaoxia Cai. "Defensive compressive time delay estimation using information bottleneck." *IEEE Signal Processing Letters 28 (2021): 1968-1972.*

Appendix A

Appendix for Chapter 3

A.1 Source Coding

Source Code: A source code C is a mapping from the range of a random variable or a set of random variables to finite length strings of symbols from a K-ary alphabet.

Expected length of a source code denoted by $L(C)$ is given as follows: $L(C) = \sum_{x\in\mathcal{X}} p(x)l(x)$, where $l(x)$ is the length of codeword $c(x)$ for a symbol $x \in \mathcal{X}$, and $p(x)$ is the probability of the symbol.

Intuitively, a good code should preserve the information content of an outcome. Since information content depends on the probability of the outcome (it is higher if probability is lower, or equivalently if the outcome is very uncertain), a good codeword will use fewer bits to encode a certain or high probability outcome and more bits to encode a low probability outcome. Thus, we expect that the smallest expected code length should be related to the average uncertainty of the random variable, i.e., the entropy.

Source coding theorem states that entropy is the fundamental limit of data compression; i.e., $\forall C : L(C) \geq H(X)$. Instead of encoding individual symbols, we can also encode blocks of symbols together. A length n block code encodes n length strings of symbols together and is denoted by $C\left(x_1, \ldots, x_n\right) =: C\left(x^n\right)$.

Proof. Consider the optimization problem:

$$\min_{l(x)} \sum_{x\in\mathcal{X}} p(x)l(x) \quad \text{such that} \quad \sum_{x\in\mathcal{X}} 2^{-l(x)} \leq 1$$

The above finds the shortest possible code length subject to satisfying the Kraft inequality. If we relax the the codelengths to be non-integer, then we can obtain a lower bound. To do this, the Lagrangian is:

$$\mathcal{L} = \sum_{x\in\mathcal{X}} p(x)l(x) + \lambda \left(\sum_{x\in\mathcal{X}} 2^{-l(x)} - 1 \right)$$

Taking derivatives with respect to $l(x)$ and λ and setting to 0, leading to:

$$p(x) + \ln 2\lambda 2^{-l(x)} = 0$$
$$\sum_{x\in\mathcal{X}} 2^{-l(x)} - 1 = 0$$

Solving this for $l(x)$ leads to $l(x) = \log \frac{1}{p(x)}$, which can be verified by direct substitution. This proves the lower bound. □

Theoretical analysis can be seen in [27].

A.2 Maximum Data Rate

We first introduce the Nyquist ISI criterion:

[Nyquist ISI criterion] If we denote the channel impulse response as $h(t)$, then the condition for an ISI-free response can be expressed as:

$$h\left(nT_s\right) = \begin{cases} 1; & n = 0 \\ 0; & n \neq 0 \end{cases}$$

for all integers n, where T_s is the symbol period. The Nyquist ISI criterion says that this is equivalent to:

$$\frac{1}{T_s}\sum_{k=-\infty}^{+\infty} H\left(f-\frac{k}{T_s}\right)=1 \quad \forall f$$

where $H(f)$ is the Fourier transform of $h(t)$.

We may now state the Nyquist ISI criterion for distortionless baseband transmission in the absence of noise: The frequency function $H(f)$ eliminates intersymbol interference for samples taken at interval T_s provide that it satisfies Equation A.2.

The simplest way of satisfing Equation A.2 is to specify the frequency function $H(f)$ to be in the form of a rectangular function, as showing by

$$\begin{aligned} H(f) &= \begin{cases} \frac{1}{2\mathbb{W}}, & -W < f < \mathbb{W} \\ 0, & |f| > \mathbb{W} \end{cases} \\ &= \frac{1}{2W}\operatorname{rect}\left(\frac{f}{2W}\right) \end{aligned}$$

where rect(f) stands for a rectangular function of unit amplitude and unit support centered on $f = 0$, and the overall system bandwidth W is definded by

$$W = \frac{1}{2T_s} = \frac{R_b}{2}$$

The special value of the bit rate $R_b = 2W$ is called the Nyquist rate, and W is itself called the Nyquist bandwidth.

The key here is that we have restricted ourselves to binary transmission and are limited to $2W$ bits/s no matter how much we increase the signal-to-noise ratio. The way to attain a higher R_b value is to replace the binary transmission system with a multilevel system, often termed an K-ary transmission system, with $K > 2$. An K-ary channel can pass $2Wlog_2K$ bits/s with an acceptable error rate.

Thus, we conclude that the bit rate $R_b \leq R_{max} = 2Wlog_2K$.

A.3 Informative Multi-agent Communication Algorithm

For completeness, we provide the IMAC algorithm below.

Algorithm 3: Informative Multi-agent Communication

```
Initialize the network parameters $\theta_a$, $\theta_{pro}$, $\theta_Q$, and $\phi_{sch}$
Initialize the target network parameters $\theta'_a$, and $\theta'_Q$
for episode ← 1 to num_episodes do
    Reset the environment for $t \leftarrow 1$ to num_step do
        Get features $h_i = [o_i, c_i]$ for each agent $i$
        Each agent $i$ gets messages from channel $m_i = \pi^i_{pro}(h_i)$
        Get scheduled message $c_i = f_{sch}(m_1, \cdots, m_n)$
        Each agent $i$ selects action based on features and messages
          $a_i = \pi^i_a(h_i, c_i)$
        Execute actions $a = (a_1, \cdots, a_n)$, and observe reward $r$ new
          observation $o_i$ for each agent $i$
        Store $(\boldsymbol{o_t}, a, r, \boldsymbol{o_{t+1}}, \boldsymbol{m}, \boldsymbol{c})$ in replay buffer $D$
        if episode%update threshold == 0 then
            Sample a random mini-batch of $S$ samples $(\boldsymbol{o}, a, r, \boldsymbol{o'}, \boldsymbol{m}, \boldsymbol{c})$ from $D$
            Obtain the features $h'_i = [o'_i, c_i]$ and the messages $m_i$ for each agent
              $i$
            Set $y^j = r^j_i + \gamma Q^{\boldsymbol{\pi'}}_i(\boldsymbol{o}, a_1', \cdots, a_n')|_{a_k'=\pi^i_a{}'(h'_i, c_i)}$
            Update Critic by minimizing the loss
              $L(\theta) = \frac{1}{S}\sum_j (Q(\boldsymbol{o}, a_1, \cdots, a_n) - \hat{y})^2$
            Update policy, protocol and scheduler using the sampled policy
              gradients $\nabla_{\theta_i} \tilde{J}(\pi_i)$ for each agent $i$
            Update all target networks' parameters for each agent $i$:
              $\theta_i' = \tau\theta_i + (1-\tau)\theta_i'$
        end
    end
end
```

A.4 The Information Bottleneck Method

A.4.1 Background

The information bottleneck method provides a principled way to extract information that is present in one variable that is relevant for predicting another variable.

Consider X and Y respectively as the input source and target, and let Z be an internal representation, i.e., a stochastic encoding, of any hidden layer of the network, defined by a parametric encoder $p(\mathbf{z}|\mathbf{x};\boldsymbol{\theta})$. The goal is to learn an encoding that is maximally informative about the target Y, which is measured by the mutual information between Y and Z, where

$$I(Z,Y;\boldsymbol{\theta}) = \int p(z,y|\boldsymbol{\theta}) \log \frac{p(z,y|\boldsymbol{\theta})}{p(z|\boldsymbol{\theta})p(y|\boldsymbol{\theta})} dxdy \tag{A.1}$$

Notice that taking the identity encoding always ensures a maximally informative representation if only with the above objective, but it is not a useful representation obviously. It is evident to constrain on encoding's complexity if we want the best representation, i.e., Z. [60] proposed the information bottleneck that expresses the trade-off between the mutual information measures $I(X,Z)$ and $I(Z,Y)$. This suggests the objective:

$$\max_{\boldsymbol{\theta}} \; I(Z,Y;\boldsymbol{\theta}) \qquad s.t. \; I(X,Z;\boldsymbol{\theta}) \leq I_c \tag{A.2}$$

where I_c is the information constraint. Equivalently, with the introduction of a Lagrange multiplier β we can maximize the objective function:

$$L(\boldsymbol{\theta}) = I(Z,Y;\boldsymbol{\theta}) - \beta I(X,Z;\boldsymbol{\theta}) \tag{A.3}$$

where β controls the trade-off. Intuitively, the first term encourages Z to be predictive of Y ; the second term encourages Z to "forget" X. Essentially it forces Z to act like a minimal sufficient statistic of X for predicting Y.

A.4.2 Why IMAC works?

We discuss why information bottleneck works in multi-agent communication.

We first introduce minimal sufficient statistics: a transformation $T(X)$ of the data X is a minimal sufficient statistic if $T(X) \in \arg\min_S I(X,S(X))$, where $S(X)$ is s.t. $I(\theta,S(X)) = \max_{T'} I\left(\theta,T'(X)\right)$

Information bottleneck principle generalizes the notion of minimal sufficient statistics and suggests using a summary of the data $T(X)$ that has least mutual information with the data X while preserving some amount of information about an auxiliary variable Y.

According to [154], from a learning perspective, we discuss the role of $I(X;T)$, the compression or minimality term in information bottleneck, as a regularizer when maximizing $I(Y;T)$.

Reinforcement learning learns a optimal action given a state. If we know the optimal action in advance, like imitation learning, then we would maximize the mutual information between the state and its corresponding optimal action, which is a straightforward application of supervised learning. Here, X represents the states, T represents the messages, Y represents the actions. Note that without regularization, $I(Y;T)$ can be maximized by setting $T = X$. However, $p(x|y)$ cannot be estimated efficiently from a sample of a reasonable size; It means that more samples are needed in reinforcement learning. In another words, methods with regularization on $I(X;T)$, e.g., IMAC, can accelerate convergence.

A.5 Experimental Details and Results

A.5.1 Cooperative navigation

In this environment, agents must cooperate through physical actions to reach a set of landmarks. Agents observe the relative positions of other agents and landmarks, and are collectively rewarded based on the proximity of any agent to each landmark. In other words, the agents have to 'cover' all of the landmarks. Further, the agents occupy significant physical space and are penalized when colliding with each other. Our agents learn to infer the landmark they must cover, and move there while avoiding other agents.

Training details We set the number of agents as 3,5,10 respectively. We use MLP with hidden layer size of 64 as basic module as before communication model, after communication model. We use the ADAM as optimizer with learning rate of 0.01. Since the environment of cooperative navigation does not send "terminal/done" to agents, we set each episode with a maximal steps of 25. The reward of each agent is identical, which equals to the sum of distances between agents to their nearest landmark. It means that agents are required not only to approach its nearest landmark, but also share information with each other for a common goal. We use the same hyper-parameter as MADDPG of openAI's version.

Table A.1 shows that MADDPG without communication tends to use high-entropy messages, while IMAC can convey low-entropy messages. Combined with the performance in Figure 4, we can see that under limited-bandwidth constraint, IMAC learns informative communication protocols.

A.5.2 Predator and prey

In this variant of the classic predator-prey game, some slower cooperating agents must chase some faster adversaries around a randomly generated environment with some large landmarks impeding the way. Each time the cooperative agents collide with some adversaries, the agents are rewarded while the adversary is penalized. Agents observe the relative positions and velocities of the agents, and the positions of the landmarks.

Training details We set the number of predators as 4, the number of preys as 2, and the number of landmarks as 2. We use the same architecture and hyper-parameter as configuration in cooperative navigation. We trained our agents by self-play for 100,000 episodes and then evaluate performance by cross-comparing between IMAC and the baselines. We average the episode rewards across 1000 rounds (episodes) as scores.

$sigma^2$	None	1	5	10
$H(M_i) = \frac{1}{2}\log(2\pi e\sigma^2)$	-	1.419	2.223	2.570
maddpg w/ com	3.480+-0.042	1.530+-0.038	3.147+-0.088	3.891+-0.059
IMAC train w/ bw=1	0.244+-0.028	-	-	-
IMAC train w/ bw=5	2.227+-0.002	1.383+-0.003	-	-
IMAC train w/ bw=10	2.763+-0.215	1.695+-0.044	3.017+-0.026	-

Table A.1: Entropy of messages in different limited bandwidths (number in cell represents
$H(M_i) = \frac{1}{2}\log(2\pi e\sigma^2)$, which is calculated based on running variance).

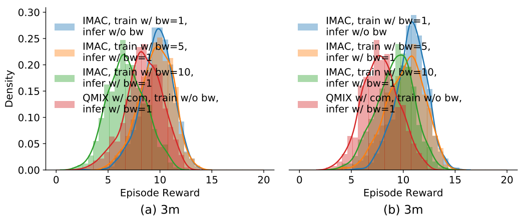


FIGURE A.1: Density plot of episode reward per agent during the execution stage. (a), (b) Reward distribution of IMAC trained with different prior distributions against MADDPG with communication under the same bandwidth constraint in 3m and 8m respectively. "bw=δ" means in the implementation of the limited bandwidth constraint, the variance $\mathbf{\Sigma}$ of Gaussian distribution is δ.

A.5.3 StarCraft II

We use the StarCraft Multi-Agent Challenge (SMAC) environment [40]. SMAC uses the StarCraft II Learning Environment to introduce a cooperative MARL environment, which focus solely on micromanagement. Each unit is controlled by an independent agent that conditions only on local observations restricted to a limited field of view centred on that unit.

We choose two scenarios: 3m and 8m as our testbeds, where a group of marines need to defeat an enemy marines. The feature vector observed by each agent contains the following attributes for both allied and enemy units within the sight range: distance, relative x, relative y, health, shield, and unit type. The discrete set of actions which agents are allowed to take consists of move in four directions, attack an enemy, stop and no-op. Dead agents can only take no-op action while live agents cannot. Some positive (negative) rewards after having enemy (allied) units killed and/or a positive (negative) bonus for winning (losing) the battle.

Training details We use QMIX architecture [41] as our algorithm backbone, where a DRQN with a recurrent layer composed of a 64-dimensional GRU and a fully-connected layer before and after. We extend QMIX with communication module. Each agent can receive other's messages from previous timestep. We use the RMSProp optimizer with learning rate $= 0.0005$ $\alpha = 0.99$, and $\epsilon = 0.00001$. The replay buffer contains the most recent 5000 episodes.We sample batches of 120 episodes uniformly from the replay buffer, and train on fully unrolled episodes, performing a single gradient descent step after every episode. The target networks are updated after every 200 training episodes.

Results Figure A.1 shows the density plot of episode reward per agent during the execution stage. We first respectively train IMAC with different prior distributions $z(M_i)$ of $N(0, 1)$, $N(0, 5)$, and $N(0, 10)$, to satisfy a limited bandwidth with the variance of 1. In the execution stage, we constrain these algorithms into different bandwidths. As depicted in Figure A.1 (a) and (b), QMIX+IMAC with different prior distributions outperform QMIX with communication in the limited bandwidth environment. Results here demonstrate that IMAC discards useless information without impairment on performance.

Appendix B

Appendix for Chapter 5

B.1 Contextual Bandit for Limited Number of Contexts

Algorithm 4: A contextual bandit algorithm for a small number of contexts

```
[1] Initialization: For each context $x$, create an instance Exp3$_x$ of algorithm Exp3 each round Invoke algorithm Exp3$_x$ with $x = x_t$ Play the action chosen by Exp3$_x$ Return reward $r_t$ to Exp3$_x$
```

B.2 Proof of Theorem 6.3.2.2

*

Proof. Let S_m be the ϵ-uniform mesh on $[0, 1]$, that is, the set of all points in $[0, 1]$ that are integer multiples of ϵ. We take $\epsilon = 1/(d-1)$ where the integer d is the number of points in S_m, which will be adjusted later in the analysis.

We apply Alg. 4 to the context space S_m. Let $f_{S_m}(x)$ be a mapping from context x to the closest point in S_m:

$$f_{S_m}(x) = \min\left(\operatorname*{argmin}_{x' \in S_m} |x - x'|\right)$$

In each round t, we replace the context x_t with $f_{S_m}(x_t)$ and call Exp3_S. The regret bound will have two components: the regret bound for Exp3_S and (a suitable notion of) the discretization error. Formally, let us define the "discretized best response" $\pi^*_{S_m} : \mathcal{X} \to \Phi$: $\pi^*_{S_m}(x) = \pi^*\left(f_{S_m}(x)\right)$ for each context $x \in \mathcal{X}$.

We define the total reward of an algorithm Alg is $\text{Reward}\,(\text{Alg}) = \sum_{t=1}^{T} r_t$. Then the regret of Exp3_S and the discretization error are defined as:

$$\begin{aligned} R_S(T) &= \text{Reward}\,(\pi^*_S) - \text{Reward}\,(\text{Exp3}_S) \\ \text{DE}(S) &= \text{Reward}\,(\pi^*) - \text{Reward}\,(\pi^*_S)\,. \end{aligned}$$

It follows that regret is the sum $R(T) = R_S(T) + \text{DE}(S)$. We have $\mathbb{E}\left[R_S(T)\right] = \mathcal{O}(\sqrt{TK \log K})$ from Lemma 6.1, so it remains to upper bound the discretization error and adjust the discretization step ϵ.

For each round t and the respective context $x = x_t$, $r\left(\pi^*_S(x) \mid f_S(x)\right) \geq r\left(\pi^*(x) \mid f_S(x)\right) \geq r\left(\pi^*(x) \mid x\right) - \epsilon L$. The first inequality is determined by the optimality of π^*_S and the second is determined by Lipschitzness. Summing this up over all rounds t, we obtain $\mathbb{E}\left[\text{Reward}\,(\pi^*_S)\right] \geq \text{Reward}\left[\pi^*\right] - \epsilon LT$.

Thus, the regret is that

$$\mathbb{E}[R(T)] \leq \epsilon LT + O\left(\sqrt{\frac{1}{\epsilon} TK \log T}\right) = O\left(T^{2/3}(LK \log T)^{1/3}\right) \tag{B.1}$$

For the last inequality, we choose $\epsilon = (\frac{K \log T}{TL^2})^{1/3}$. □

B.3 SPC on GRF 11vs11 Full Game

We also conduct experiments on the GRF 11vs11 full game scenario with sparse reward. As shown in Fig. B.1, SPC achieves about 50% win rate against built-in AI in the target task after training with 200 million timesteps. This is non-trivial as this is one of the most challenging benchmarks for MARL community, and most current MARL methods struggle to achieve progress without hand-crafted engineering.

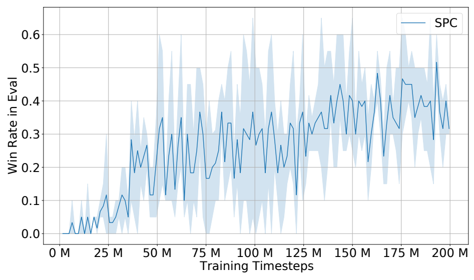


FIGURE B.1: The performance of SPC on the 11v11 scenario.

B.4 Qualitatively Analysis On Low-Level Skills

We demonstrate game statistics under different high-level actions. For example, the times of shooting, passing and running actions per game in GRF. These different low-level policies are induced by the high-level actions. We evaluate these statistics by fixing one agent's high-level actions and maintaining other agents with SPC. The results in Table B.1 are averaged over five runs in the 5vs5 scenario.

TABLE B.1: Statistics of low-level skills.

	shooting per game	passing per game	running per game
skill 1	7.9 times	0.5 times	2254 time steps
skill 2	2.3 times	26.4 times	2149 time steps
skill 3	1.6 times	3.9 times	2875 time steps

B.5 Comparing Different Teacher Algorithms on GRF Corner-5

To further illustrate the effectiveness of the SPC teacher module, we conduct experiments on the corner-5 scenario on GRF, where the target task is to control five of the eleven players to obtain a goal in the GRF Corner scenario. The experiments are designed to determine whether or not the contextual bandit in SPC outperforms alternative curriculum learning methods to schedule the number of agents in training. We compare SPC teacher against non-curriculum training (None), uniform task sampling (Uniform), a state-of-the-art curriculum learning method

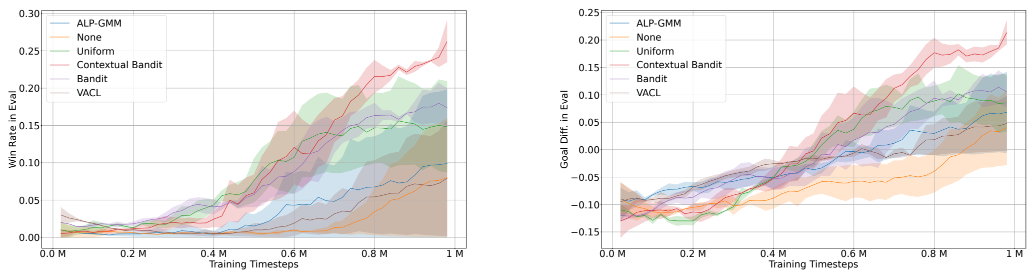
(ALP-GMM), and a multi-agent curriculum learning method (VACL). The training task space consists of n agents, where $n \in \{1, 3, 5\}$. All teachers have the same base architecture without transformer architecture and HRL. We also investigate the ablation of the RNN-based contexts (see Contextual Bandit and Bandit). Fig. B.2 shows the benefit of SPC contextual bandit over other ACL methods after training with one million timesteps.

B.6 Implementation Details

We use the default implementation of Proximal Policy Optimization (PPO) in Ray RLlib, which scales out using multiple workers for experience collection. This allows us to use a large amount of rollouts from parallel workers during training to ameliorate high variance and aid exploration. We do multiple rollouts in parallel with distributed workers and use parameter sharing for each agent. The trainer broadcasts new weights to the workers after their synchronous sampling.

B.6.1 Google Research Football

We set five tasks for training the GRF 5vs5 scenario, including 5vs5, 3vs3, Pass-Shoot, 3vs1, and Empty-Goal. In the Empty-Goal, one agent need to move forward and shoot with an empty goal. In Pass-Shoot and 3vs3, two agents are controlled to play against a goalkeeper and three players, with different position initialization. In 3vs1, three agents are controlled to play against a center-back and a goalkeeper.



(A) Win Rate (B) Goal Difference

Figure B.2: The evaluation performance of various teacher algorithms on the GRF corner-5 scenario.

In 5vs5, four agents are controlled to play against five players. Without loss of generality, we initialize all player with fixed positions and roles as center midfielders.

We use both MLP and self-attention mechanism for the high-level policy, and use MLP for the low-level policy. For high-level policy, the input is first projected to an embedding using two hidden layers with 256 units each and ReLU activation, which is then fed into multi-head self-attention (8 heads, 64 units each). The output is then projected to the actions and values using another fully connected layer with 256 units. For low-level policy, we use MLP with two hidden layers with 256 units each, i.e., the default configuration of policy network in RLlib.

Table B.2: SPC hyper-parameters.

(a) SPC hyper-parameters used in GRF.

Name	Value
Discount rate	0.99
GAE parameter	1.0
KL coefficient	0.2
Rollout fragment length	1000
Training batch size	100000
SGD minibatch size	10000
# of SGD iterations	60
Learning rate	1e-4
Entropy coefficient	0.0
Clip parameter	0.3
Value function clip parameter	10.0

(b) SPC hyper-parameters used in MPE.

Name	Value
Discount rate	0.99
GAE parameter	1.0
KL coefficient	0.5
# of SGD iterations	10
Learning rate	1e-4
Entropy coefficient	0.0
Clip parameter	0.3
Value function clip parameter	10.0

B.6.2 MPE

In MPE tasks, agents must cooperate through physical actions to reach a set of landmarks. Agents observe the relative positions of other agents and landmarks, and are collectively rewarded based on the proximity of any agent to each landmark. In other words, the agents have to cover all of the landmarks. Further, the agents are penalized when colliding with each other. The agents need to infer the landmark to cover and move there while avoid colliding with other agents.

The hyper-parameters of SPC in MPE are shown in Table B.2b. In MPE, hyper-parameters such as rollout fragment length, training batch size and SGD minibatch size are adjusted according to horizon of the scenarios so that policy are updated after episodes are done. We use the same neural network architecture as in GRF, but with 128 units for all MLP hidden layers. Other omitted hyper-parameters follow the default configuration in RLlib PPO implementation.

Bibliography

[1] Vijay Konda and John Tsitsiklis. Actor-critic algorithms. *Advances in neural information processing systems*, 12, 1999. xv, 17

[2] Natasha Jaques, Angeliki Lazaridou, Edward Hughes, Caglar Gulcehre, Pedro A Ortega, DJ Strouse, Joel Z Leibo, and Nando de Freitas. Intrinsic social motivation via causal influence in multi-agent rl. 2018. 2

[3] 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. 27

[4] Bowen Baker, Ingmar Kanitscheider, Todor Markov, Yi Wu, Glenn Powell, Bob McGrew, and Igor Mordatch. Emergent tool use from multi-agent autocurricula. *arXiv preprint arXiv:1909.07528*, 2019. 97

[5] 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. 97, 104, 105

[6] David Abel. A theory of abstraction in reinforcement learning. *arXiv preprint arXiv:2203.00397*, 2022. 2

[7] Gavin A Rummery and Mahesan Niranjan. *On-line Q-learning using connectionist systems*, volume 37. University of Cambridge, Department of Engineering Cambridge, UK, 1994. 11

[8] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8:279–292, 1992. 11

[9] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013. 12, 47, 65

[10] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8:229–256, 1992. 14

[11] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897, 2015. 15

[12] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017. 16, 96

[13] 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. 18, 47, 65, 73

[14] 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. 19

[15] Tonghan Wang, Jianhao Wang, Chongyi Zheng, and Chongjie Zhang. Learning nearly decomposable value functions via communication minimization. In *International Conference on Learning Representations*, 2020. URL `https://openreview.net/forum?id=HJx-3grYDB`. 23, 97

[16] 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. 23, 26, 35

[17] Sainbayar Sukhbaatar, Rob Fergus, et al. Learning multiagent communication with backpropagation. In *Advances in Neural Information Processing Systems*, pages 2244–2252, 2016. 26, 97

[18] Peng Peng, Quan Yuan, Ying Wen, Yaodong Yang, Zhenkun Tang, Haitao Long, and Jun Wang. Multiagent bidirectionally-coordinated nets for learning to play starcraft combat games. *arXiv preprint arXiv:1703.10069*, 2, 2017. 23, 26

[19] Jiechuan Jiang and Zongqing Lu. Learning attentional communication for multi-agent cooperation. In *Advances in Neural Information Processing Systems*, pages 7265–7275, 2018. 23, 26, 27, 39, 97

[20] Ozsel Kilinc and Giovanni Montana. Multi-agent deep reinforcement learning with extremely noisy observations. *arXiv preprint arXiv:1812.00922*, 2018. 26

[21] Abhishek Das, Théophile Gervet, Joshua Romoff, Dhruv Batra, Devi Parikh, Mike Rabbat, and Joelle Pineau. TarMAC: Targeted multi-agent communication. In *Proceedings of the 36th International Conference on Machine Learning*, pages 1538–1546, 2019. 26, 37, 97

[22] Amanpreet Singh, Tushar Jain, and Sainbayar Sukhbaatar. Learning when to communicate at scale in multiagent cooperative and competitive tasks. *arXiv preprint arXiv:1812.09755*, 2018. 23, 26, 97

[23] Ryan Lowe, Jakob Foerster, Y-Lan Boureau, Joelle Pineau, and Yann Dauphin. On the pitfalls of measuring emergent communication. *arXiv preprint arXiv:1903.05168*, 2019. 24, 39

[24] 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, 2013. 25, 27

[25] Daewoo Kim, Sangwoo Moon, David Hostallero, Wan Ju Kang, Taeyoung Lee, Kyunghwan Son, and Yung Yi. Learning to schedule communication in multi-agent reinforcement learning. *arXiv preprint arXiv:1902.01554*, 2019. 25, 26, 27, 35, 37, 97

[26] Hangyu Mao, Zhibo Gong, Zhengchao Zhang, Zhen Xiao, and Yan Ni. Learning multi-agent communication under limited-bandwidth restriction for internet packet routing. *arXiv preprint arXiv:1903.05561*, 2019. 25, 26, 27, 37

[27] Claude Elwood Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27(3):379–423, 1948. 25, 31, 110

[28] R.L. Freeman. *Telecommunication System Engineering*, pages 398–399. Wiley Series in Telecommunications and Signal Processing. Wiley, 2004. ISBN 9780471451334. 25, 29, 31

[29] Naftali Tishby, Fernando C Pereira, and William Bialek. The information bottleneck method. *arXiv preprint physics/0004057*, 2000. 25

[30] Alexander A Alemi, Ian Fischer, Joshua V Dillon, and Kevin Murphy. Deep variational information bottleneck. *arXiv preprint arXiv:1612.00410*, 2016. 25, 33, 34, 55

[31] Angeliki Lazaridou, Alexander Peysakhovich, and Marco Baroni. Multi-agent cooperation and the emergence of (natural) language. *arXiv preprint arXiv:1612.07182*, 2016. 26

[32] Igor Mordatch and Pieter Abbeel. Emergence of grounded compositional language in multi-agent populations. In *AAAI Conference on Artificial Intelligence*, 2018. 26

[33] Xue Bin Peng, Angjoo Kanazawa, Sam Toyer, Pieter Abbeel, and Sergey Levine. Variational discriminator bottleneck: Improving imitation learning, inverse RL, and gans by constraining information flow. *arXiv preprint arXiv:1810.00821*, 2018. 27

[34] Anirudh Goyal, Riashat Islam, Daniel Strouse, Zafarali Ahmed, Matthew Botvinick, Hugo Larochelle, Sergey Levine, and Yoshua Bengio. Infobot: Transfer and exploration via the information bottleneck. *arXiv preprint arXiv:1901.10902*, 2019. 27

[35] Claudia V Goldman and Shlomo Zilberstein. Optimizing information exchange in cooperative multi-agent systems. In *Proceedings of the second international joint conference on Autonomous agents and multiagent systems*, pages 137–144, 2003. 28

[36] Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, OpenAI 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. 28, 29, 36, 37, 89, 97

[37] Thomas M Cover and Joy A Thomas. *Elements of Information Theory*. John Wiley & Sons, 2012. 32, 55

[38] Edwin T Jaynes. Information theory and statistical mechanics. *Physical Review*, 106(4):620, 1957. 32

[39] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015. 36

[40] Mikayel Samvelyan, Tabish Rashid, Christian Schroeder de Witt, Gregory Farquhar, Nantas Nardelli, Tim G. J. Rudner, Chia-Man Hung, Philiph H. S. Torr, Jakob Foerster, and Shimon Whiteson. The StarCraft Multi-Agent Challenge. *CoRR*, abs/1902.04043, 2019. 37, 42, 116

[41] Tabish Rashid, Mikayel Samvelyan, Christian Schroeder De Witt, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. Qmix: monotonic value function factorisation for deep multi-agent reinforcement learning. *arXiv preprint arXiv:1803.11485*, 2018. 37, 99, 116

[42] Rundong Wang, Runsheng Yu, Bo An, and Zinovi Rabinovich. I2hrl: Interactive influence-based hierarchical reinforcement learning. In *Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence*, pages 3131–3138, 2021. 47, 97

[43] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484, 2016. 47, 65

[44] Ofir Nachum, Shixiang Shane Gu, Honglak Lee, and Sergey Levine. Data-efficient hierarchical reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 3303–3313, 2018. 48, 49, 51, 59, 97

[45] Andrew Levy, Robert Platt, and Kate Saenko. Hierarchical reinforcement learning with hindsight. In *International Conference on Learning Representations*, 2019. `https://openreview.net/forum?id=ryzECoAcY7`. 48, 49, 59

[46] Marc Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Remi Munos. Unifying count-based exploration and intrinsic motivation. In *Advances in Neural Information Processing Systems*, pages 1471–1479, 2016. 48

[47] Benjamin Eysenbach, Abhishek Gupta, Julian Ibarz, and Sergey Levine. Diversity is all you need: Learning skills without a reward function. *arXiv preprint arXiv:1802.06070*, 2018. 48, 50, 96

[48] Yan Duan, Xi Chen, Rein Houthooft, John Schulman, and Pieter Abbeel. Benchmarking deep reinforcement learning for continuous control. In *International Conference on Machine Learning*, pages 1329–1338, 2016. 49, 58

[49] Pierre-Luc Bacon, Jean Harb, and Doina Precup. The option-critic architecture. In *AAAI Conference on Artificial Intelligence*, pages 1726–1734, 2017. 49, 96

[50] George Konidaris and Andrew Barto. Efficient skill learning using abstraction selection. In *International Joint Conference on Artificial Intelligence*, pages 1107–1112, 2009. 49

[51] 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, 2017. 49

[52] Carlos Florensa, Yan Duan, and Pieter Abbeel. Stochastic neural networks for hierarchical reinforcement learning. *arXiv preprint arXiv:1704.03012*, 2017. 50

[53] Nicolas Heess, Greg Wayne, Yuval Tassa, Timothy Lillicrap, Martin Riedmiller, and David Silver. Learning and transfer of modulated locomotor controllers. *arXiv preprint arXiv:1610.05182*, 2016. 50

[54] Ofir Nachum, Haoran Tang, Xingyu Lu, Shixiang Gu, Honglak Lee, and Sergey Levine. Why does hierarchy (sometimes) work so well in reinforcement learning? *arXiv preprint arXiv:1909.10618*, 2019. 50, 56

[55] Tejas D Kulkarni, Karthik Narasimhan, Ardavan Saeedi, and Josh Tenenbaum. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. In *Advances in Neural Information Processing Systems*, pages 3675–3683, 2016. 50

[56] Jacob Rafati and David C Noelle. Efficient exploration through intrinsic motivation learning for unsupervised subgoal discovery in model-free hierarchical reinforcement learning. *arXiv preprint arXiv:1911.10164*, 2019. 50

[57] Frans Oliehoek, Stefan Witwicki, and Leslie Kaelbling. Influence-based abstraction for multiagent systems. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 26, pages 1422–1428, 2012. 50

[58] Georgios Papoudakis, Filippos Christianos, Arrasy Rahman, and Stefano V Albrecht. Dealing with non-stationarity in multi-agent deep reinforcement learning. *arXiv preprint arXiv:1906.04737*, 2019. 50

[59] Neil C Rabinowitz, Frank Perbet, H Francis Song, Chiyuan Zhang, SM Eslami, and Matthew Botvinick. Machine theory of mind. *arXiv preprint arXiv:1802.07740*, 2018. 53

[60] Naftali Tishby and Noga Zaslavsky. Deep learning and the information bottleneck principle. In *2015 IEEE Information Theory Workshop (ITW)*, pages 1–5, 2015. 55, 113

[61] Youngjin Kim, Wontae Nam, Hyunwoo Kim, Ji-Hoon Kim, and Gunhee Kim. Curiosity-bottleneck: Exploration by distilling task-specific novelty. In *International Conference on Machine Learning*, pages 3379–3388, 2019. 57

[62] Scott Fujimoto, Herke van Hoof, and David Meger. Addressing function approximation error in actor-critic methods. *arXiv preprint arXiv:1802.09477*, 2018. 59

[63] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(Nov):2579–2605, 2008. 61

[64] Rundong Wang, Hongxin Wei, Bo An, Zhouyan Feng, and Jun Yao. Commission fee is not enough: A hierarchical reinforced framework for portfolio management. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 626–633, 2021. 65

[65] Amir Mosavi, Pedram Ghamisi, Yaser Faghan, and Puhong Duan. Comprehensive review of deep reinforcement learning methods and applications in economics. *arXiv preprint arXiv:2004.01509*, 2020. 65

[66] Jing Zhang and Dacheng Tao. Empowering things with intelligence: A survey of the progress, challenges, and opportunities in artificial intelligence of things. *IEEE Internet of Things Journal*, 2020. doi: 10.1109/JIOT.2020.3039359. 65

[67] Saud Almahdi and Steve Y Yang. An adaptive portfolio trading system: A risk-return portfolio optimization using recurrent reinforcement learning with expected maximum drawdown. *Expert Systems with Applications*, 87:267–279, 2017. 65

[68] Zhengyao Jiang and Jinjun Liang. Cryptocurrency portfolio management with deep reinforcement learning. In *2017 Intelligent Systems Conference (IntelliSys)*, pages 905–913. IEEE, 2017. 65

[69] Lili Tang. An actor-critic-based portfolio investment method inspired by benefit-risk optimization. *Journal of Algorithms & Computational Technology*, 12(4):351–360, 2018. 66

[70] Pengqian Yu, Joon Sern Lee, Ilya Kulyatin, Zekun Shi, and Sakyasingha Dasgupta. Model-based deep reinforcement learning for dynamic portfolio optimization. *arXiv preprint arXiv:1901.08740*, 2019. 66

[71] Zhipeng Liang, Hao Chen, Junhao Zhu, Kangkang Jiang, and Yanran Li. Adversarial deep reinforcement learning in portfolio management. *arXiv preprint arXiv:1808.09940*, 2018. 66, 73

[72] Yunan Ye, Hengzhi Pei, Boxin Wang, Pin-Yu Chen, Yada Zhu, Jun Xiao, and Bo Li. Reinforcement-learning based portfolio management with augmented asset movement prediction states. *arXiv preprint arXiv:2002.05780*, 2020. 66

[73] Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*, pages 1057–1063, 2000. 74

[74] Arash Tavakoli, Fabio Pardo, and Petar Kormushev. Action branching architectures for deep reinforcement learning. In *Thirty-Second AAAI Conference on Artificial Intelligence*, pages 4131–4138, 2018. 74

[75] Yuriy Nevmyvaka, Yi Feng, and Michael Kearns. Reinforcement learning for optimized trade execution. In *Proceedings of the 23rd International Conference on Machine Learning*, pages 673–680, 2006. 75

[76] Thomas M Cover. Universal portfolios. In *The Kelly Capital Growth Investment Criterion: Theory and Practice*, pages 181–209. World Scientific, 2011. 77

[77] Bin Li and Steven CH Hoi. On-line portfolio selection with moving average reversion. In *Proceedings of the 29th International Coference on International Conference on Machine Learning*, pages 563–570, 2012. 77

[78] Li Gao and Weiguo Zhang. Weighted moving average passive aggressive algorithm for online portfolio selection. In *2013 5th International Conference on Intelligent Human-Machine Systems and Cybernetics*, volume 1, pages 327–330. IEEE, 2013. 77

[79] Alexei A Gaivoronski and Fabio Stella. Stochastic nonstationary optimization for finding universal portfolios. *Annals of Operations Research*, 100(1-4):165–188, 2000. 77

[80] Allan Borodin, Ran El-Yaniv, and Vincent Gogan. Can we learn to beat the best stock. In *Advances in Neural Information Processing Systems*, pages 345–352, 2004. 77

[81] Zhengyao Jiang, Dixing Xu, and Jinjun Liang. A deep reinforcement learning framework for the financial portfolio management problem. *arXiv preprint arXiv:1706.10059*, 2017. 77

[82] Rundong Wang, Longtao Zheng, Wei Qiu, Bowei He, Bo An, Zinovi Rabinovich, Yujing Hu, Yingfeng Chen, Tangjie Lv, and Changjie Fan. Towards skilled population curriculum for multi-agent reinforcement learning. *arXiv preprint arXiv:2302.03429*, 2023. 87

[83] Kaiqing Zhang, Zhuoran Yang, and Tamer Başar. Multi-agent reinforcement learning: A selective overview of theories and algorithms. *Handbook of Reinforcement Learning and Control*, pages 321–384, 2021. 88

[84] Yaodong Yang and Jun Wang. An overview of multi-agent reinforcement learning from game theoretical perspective. *arXiv preprint arXiv:2011.00583*, 2020. 88

[85] Yujing Hu, Weixun Wang, Hangtian Jia, Yixiang Wang, Yingfeng Chen, Jianye Hao, Feng Wu, and Changjie Fan. Learning to utilize shaping rewards: A new approach of reward shaping. *arXiv preprint arXiv:2011.02669*, 2020. 88

[86] Jiayu Chen, Yuanxin Zhang, Yuanfan Xu, Huimin Ma, Huazhong Yang, Jiaming Song, Yu Wang, and Yi Wu. Variational automatic curriculum learning for sparse-reward cooperative multi-agent problems. *Advances in Neural Information Processing Systems*, 34, 2021. 88, 99

[87] Shiyu Huang, Wenze Chen, Longfei Zhang, Ziyang Li, Fengming Zhu, Deheng Ye, Ting Chen, and Jun Zhu. Tikick: Toward playing multi-agent football full games from single-agent demonstrations. *arXiv preprint arXiv:2110.04507*, 2021. 88

[88] Weixun Wang, Tianpei Yang, Yong Liu, Jianye Hao, Xiaotian Hao, Yujing Hu, Yingfeng Chen, Changjie Fan, and Yang Gao. From few to more: Large-scale dynamic multiagent curriculum learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 7293–7300, 2020. 88

[89] Qian Long, Zihan Zhou, Abhibav Gupta, Fei Fang, Yi Wu, and Xiaolong Wang. Evolutionary population curriculum for scaling multi-agent reinforcement learning. *arXiv preprint arXiv:2003.10423*, 2020. 88

[90] Jiachen Yang, Igor Borovikov, and Hongyuan Zha. Hierarchical cooperative multi-agent reinforcement learning with skill discovery. *arXiv preprint arXiv:1912.03558*, 2019. 88

[91] Alexander Sasha Vezhnevets, Yuhuai Wu, Remi Leblond, and Joel Z Leibo. Options as responses: Grounding behavioural hierarchies in multi-agent rl. *arXiv preprint arXiv:1906.01470*, 2019. 88

[92] Tonghan Wang, Tarun Gupta, Anuj Mahajan, Bei Peng, Shimon Whiteson, and Chongjie Zhang. Rode: Learning roles to decompose multi-agent tasks. *arXiv preprint arXiv:2010.01523*, 2020. 88

[93] Zhen-Jia Pang, Ruo-Ze Liu, Zhou-Yu Meng, Yi Zhang, Yang Yu, and Tong Lu. On reinforcement learning for full-length game of starcraft. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2019. 88

[94] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997. 89

[95] 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. 89, 95

[96] 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. 89, 98

[97] Peter Auer, Nicolo Cesa-Bianchi, Yoav Freund, and Robert E Schapire. The nonstochastic multiarmed bandit problem. *SIAM Journal on Computing*, 32 (1):48–77, 2002. 90, 92, 94

[98] Elad Hazan and Nimrod Megiddo. Online learning with prior knowledge. In *International Conference on Computational Learning Theory*, pages 499–513. Springer, 2007. 90

[99] Tian Zhang, Raghu Ramakrishnan, and Miron Livny. Birch: An efficient data clustering method for very large databases. *ACM Aigmod Record*, 25 (2):103–114, 1996. 92, 95

[100] Heikki Hyötyniemi. Turing machines are recurrent neural networks. In *STeP '96/Publications of the Finnish Artificial Intelligence Society*, 1996. 93

[101] Shariq Iqbal, Christian A Schroeder De Witt, Bei Peng, Wendelin Böhmer, Shimon Whiteson, and Fei Sha. Randomized entity-wise factorization for multi-agent reinforcement learning. In *International Conference on Machine Learning*, pages 4596–4606. PMLR, 2021. 95

[102] Siyi Hu, Fengda Zhu, Xiaojun Chang, and Xiaodan Liang. Updet: Universal multi-agent reinforcement learning via policy decoupling with transformers. *arXiv preprint arXiv:2101.08001*, 2021. 95

[103] Wei Fu, Chao Yu, Zelai Xu, Jiaqi Yang, and Yi Wu. Revisiting some common practices in cooperative multi-agent reinforcement learning. *arXiv preprint arXiv:2206.07505*, 2022. 96

[104] Rémy Portelas, Cédric Colas, Lilian Weng, Katja Hofmann, and Pierre-Yves Oudeyer. Automatic curriculum learning for deep RL: A short survey. *arXiv preprint arXiv:2003.04664*, 2020. 97

[105] Sanmit Narvekar and Peter Stone. Learning curriculum policies for reinforcement learning. *arXiv preprint arXiv:1812.00285*, 2018. 97

[106] Ilge Akkaya, Marcin Andrychowicz, Maciek Chociej, Mateusz Litwin, Bob McGrew, Arthur Petron, Alex Paino, Matthias Plappert, Glenn Powell, Raphael Ribas, et al. Solving rubik's cube with a robot hand. *arXiv preprint arXiv:1910.07113*, 2019. 97

[107] Bhairav Mehta, Manfred Diaz, Florian Golemo, Christopher J Pal, and Liam Paull. Active domain randomization. In *Conference on Robot Learning*, pages 1162–1176. PMLR, 2020. 97

[108] Rémy Portelas, Cédric Colas, Katja Hofmann, and Pierre-Yves Oudeyer. Teacher algorithms for curriculum learning of deep RL in continuously parameterized environments. In *Conference on Robot Learning*, pages 835–853, 2020. 97

[109] Lu Jiang, Deyu Meng, Qian Zhao, Shiguang Shan, and Alexander G Hauptmann. Self-paced curriculum learning. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015. 97

[110] Carlos Florensa, David Held, Xinyang Geng, and Pieter Abbeel. Automatic goal generation for reinforcement learning agents. In *International Conference on Machine Learning*, pages 1515–1528. PMLR, 2018. 97

[111] Minqi Jiang, Edward Grefenstette, and Tim Rocktäschel. Prioritized level replay. In *International Conference on Machine Learning*, pages 4940–4950. PMLR, 2021. 97

[112] Minqi Jiang, Michael Dennis, Jack Parker-Holder, Jakob Foerster, Edward Grefenstette, and Tim Rocktäschel. Replay-guided adversarial environment design. *Advances in Neural Information Processing Systems*, 34:1884–1897, 2021. 97

[113] Pascal Klink, Carlo D'Eramo, Jan R Peters, and Joni Pajarinen. Self-paced deep reinforcement learning. *Advances in Neural Information Processing Systems*, 33:9216–9227, 2020. 97

[114] Pascal Klink, Haoyi Yang, Carlo D'Eramo, Jan Peters, and Joni Pajarinen. Curriculum reinforcement learning via constrained optimal transport. In *International Conference on Machine Learning*, pages 11341–11358. PMLR, 2022. 97

[115] Maxwell Svetlik, Matteo Leonetti, Jivko Sinapov, Rishi Shah, Nick Walker, and Peter Stone. Automatic curriculum graph generation for reinforcement learning agents. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 2017. 97

[116] Wolfgang Banzhaf, Bert Baumgaertner, Guillaume Beslon, René Doursat, James A Foster, Barry McMullin, Vinicius Veloso De Melo, Thomas Miconi, Lee Spector, Susan Stepney, et al. Defining and simulating open-ended novelty: Requirements, guidelines, and challenges. *Theory in Biosciences*, 135 (3):131–161, 2016. 97

[117] Joel Lehman, Kenneth O Stanley, et al. Exploiting open-endedness to solve problems through the search for novelty. In *ALIFE*, pages 329–336. Citeseer, 2008.

[118] Russell K Standish. Open-ended artificial evolution. *International Journal of Computational Intelligence and Applications*, 3(02):167–175, 2003. 97

[119] Siqi Liu, Guy Lever, Josh Merel, Saran Tunyasuvunakool, Nicolas Heess, and Thore Graepel. Emergent coordination through competition. *arXiv preprint arXiv:1902.07151*, 2019. 97

[120] Abhinav Gupta, Marc Lanctot, and Angeliki Lazaridou. Dynamic population-based meta-learning for multi-agent communication with natural language. *Advances in Neural Information Processing Systems*, 34:16899–16912, 2021. 97

[121] Rémy Portelas, Clément Romac, Katja Hofmann, and Pierre-Yves Oudeyer. Meta automatic curriculum learning. *arXiv preprint arXiv:2011.08463*, 2020. 97

[122] Joel Z Leibo, Edward Hughes, Marc Lanctot, and Thore Graepel. Autocurricula and the emergence of innovation from social interaction: A manifesto for multi-agent intelligence research. *arXiv preprint arXiv:1903.00742*, 2019. 97

[123] Shayegan Omidshafiei, Dong-Ki Kim, Miao Liu, Gerald Tesauro, Matthew Riemer, Christopher Amato, Murray Campbell, and Jonathan P How. Learning to teach in cooperative multiagent reinforcement learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 6128–6136, 2019. 97

[124] Ofir Nachum, Shixiang Gu, Honglak Lee, and Sergey Levine. Near-optimal representation learning for hierarchical reinforcement learning. *arXiv preprint arXiv:1810.01257*, 2018. 97

[125] Sainbayar Sukhbaatar, Emily Denton, Arthur Szlam, and Rob Fergus. Learning goal embeddings via self-play for hierarchical reinforcement learning. *arXiv preprint arXiv:1811.09083*, 2018.

[126] Suraj Nair and Chelsea Finn. Hierarchical foresight: Self-supervised learning of long-horizon tasks via visual subgoal generation. *arXiv preprint arXiv:1909.05829*, 2019. 97

[127] Christian Daniel, Gerhard Neumann, and Jan Peters. Hierarchical relative entropy policy search. In *Artificial Intelligence and Statistics*, pages 273–281, 2012. 97

[128] Karol Gregor, Danilo Jimenez Rezende, and Daan Wierstra. Variational intrinsic control. *arXiv preprint arXiv:1611.07507*, 2016.

[129] Tanmay Shankar and Abhinav Gupta. Learning robot skills with temporal variational inference. In *Proceedings of the 37th International Conference on Machine Learning*. JMLR. org, 2020.

[130] Archit Sharma, Shixiang Gu, Sergey Levine, Vikash Kumar, and Karol Hausman. Dynamics-aware unsupervised discovery of skills. In *International Conference on Learning Representations*, 2020. 97

[131] Crystal Chao, Maya Cakmak, and Andrea L Thomaz. Towards grounding concepts for transfer in goal learning from demonstration. In *2011 IEEE International Conference on Development and Learning (ICDL)*, volume 2, pages 1–6. IEEE, 2011. 97

[132] Karl Moritz Hermann, Felix Hill, Simon Green, Fumin Wang, Ryan Faulkner, Hubert Soyer, David Szepesvari, Wojciech Marian Czarnecki, Max Jaderberg, Denis Teplyashin, et al. Grounded language learning in a simulated 3d world. *arXiv preprint arXiv:1706.06551*, 2017. 97

[133] Devendra Singh Chaplot, Kanthashree Mysore Sathyendra, Rama Kumar Pasumarthi, Dheeraj Rajagopal, and Ruslan Salakhutdinov. Gated-attention architectures for task-oriented language grounding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018. 97

[134] Edward C Williams, Nakul Gopalan, Mine Rhee, and Stefanie Tellex. Learning to parse natural language to grounded reward functions with weak supervision. In *2018 ieee international conference on robotics and automation (icra)*, pages 4430–4436. IEEE, 2018. 97

[135] Deblina Bhattacharjee, Tong Zhang, Sabine Süsstrunk, and Mathieu Salzmann. Mult: An end-to-end multitask learning transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12031–12041, 2022. 97

[136] Coline Devin, Abhishek Gupta, Trevor Darrell, Pieter Abbeel, and Sergey Levine. Learning modular neural network policies for multi-task and multi-robot transfer. In *2017 IEEE international conference on robotics and automation (ICRA)*, pages 2169–2176. IEEE, 2017. 97

[137] Chenghao Li, Chengjie Wu, Tonghan Wang, Jun Yang, Qianchuan Zhao, and Chongjie Zhang. Celebrating diversity in shared multi-agent reinforcement learning. *arXiv preprint arXiv:2106.02195*, 2021. 98

[138] Siyang Wu, Tonghan Wang, Chenghao Li, and Chongjie Zhang. Containerized distributed value-based multi-agent reinforcement learning. *arXiv preprint arXiv:2110.08169*, 2021. 98

[139] 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. 99

[140] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008. 102

[141] OpenAI. OpenAI Five. `https://openai.com/blog/openai-five/`, 2019. Accessed March 4, 2019. 104, 105

[142] Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Misha Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling. *Advances in Neural Information Processing Systems*, 34:15084–15097, 2021. 105

[143] Michael Janner, Qiyang Li, and Sergey Levine. Offline reinforcement learning as one big sequence modeling problem. *Advances in Neural Information Processing Systems*, 34:1273–1286, 2021. 105

[144] Scott Reed, Konrad Zolna, Emilio Parisotto, Sergio Gomez Colmenarejo, Alexander Novikov, Gabriel Barth-Maron, Mai Gimenez, Yury Sulsky, Jackie Kay, Jost Tobias Springenberg, et al. A generalist agent. *arXiv preprint arXiv:2205.06175*, 2022. 105

[145] Kuang-Huei Lee, Ofir Nachum, Mengjiao Yang, Lisa Lee, Daniel Freeman, Winnie Xu, Sergio Guadarrama, Ian Fischer, Eric Jang, Henryk Michalewski, et al. Multi-game decision transformers. *arXiv preprint arXiv:2205.15241*, 2022. 105

[146] Muning Wen, Jakub Grudzien Kuba, Runji Lin, Weinan Zhang, Ying Wen, Jun Wang, and Yaodong Yang. Multi-agent reinforcement learning is a sequence modeling problem. *arXiv preprint arXiv:2205.14953*, 2022. 105

[147] Yang Gao, Huazhe Xu, Ji Lin, Fisher Yu, Sergey Levine, and Trevor Darrell. Reinforcement learning from imperfect demonstrations. *arXiv preprint arXiv:1802.05313*, 2018. 105

[148] Todd Hester, Matej Vecerik, Olivier Pietquin, Marc Lanctot, Tom Schaul, Bilal Piot, Dan Horgan, John Quan, Andrew Sendonaris, Ian Osband, et al. Deep q-learning from demonstrations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018. 105

[149] Andrei A Rusu, Sergio Gomez Colmenarejo, Caglar Gulcehre, Guillaume Desjardins, James Kirkpatrick, Razvan Pascanu, Volodymyr Mnih, Koray Kavukcuoglu, and Raia Hadsell. Policy distillation. *arXiv preprint arXiv:1511.06295*, 2015. 105

[150] Emilio Parisotto, Jimmy Lei Ba, and Ruslan Salakhutdinov. Actor-mimic: Deep multitask and transfer reinforcement learning. *arXiv preprint arXiv:1511.06342*, 2015. 105

[151] Kai-Wei Chang, Akshay Krishnamurthy, Alekh Agarwal, Hal Daumé III, and John Langford. Learning to search better than your teacher. In *International Conference on Machine Learning*, pages 2058–2066. PMLR, 2015. 105

[152] Simon Schmitt, Jonathan J Hudson, Augustin Zidek, Simon Osindero, Carl Doersch, Wojciech M Czarnecki, Joel Z Leibo, Heinrich Kuttler, Andrew Zisserman, Karen Simonyan, et al. Kickstarting deep reinforcement learning. *arXiv preprint arXiv:1803.03835*, 2018.

[153] Ikechukwu Uchendu, Ted Xiao, Yao Lu, Banghua Zhu, Mengyuan Yan, Joséphine Simon, Matthew Bennice, Chuyuan Fu, Cong Ma, Jiantao Jiao, et al. Jump-start reinforcement learning. *arXiv preprint arXiv:2204.02372*, 2022. 105

[154] Ohad Shamir, Sivan Sabato, and Naftali Tishby. Learning and generalization with the information bottleneck. *Theoretical Computer Science*, 411(29-30): 2696–2711, 2010. 114