中图分类号：TP391　　论文编号：1028716 21-S115
学科分类号：083900

硕士学位论文

多智能体强化学习协作算法研究与应用

研究生姓名　　文超
学科、专业　　网络空间安全
研 究 方 向　　多智能体强化学习
指 导 教 师　　谭晓阳 教授

南京航空航天大学
研究生院　计算机科学与技术学院

二〇二一年三月

Nanjing University of Aeronautics and Astronautics
The Graduate School
College of Computer Science and Technology

Research and Application of Cooperative Multi-Agent Reinforcement Learning

A Thesis in

Cyberspace Security

by

Chao Wen

Advised by

Prof. Xiaoyang Tan

Submitted in Partial Fulfillment

of the Requirements

for the Degree of

Master of Engineering

March, 2021

承诺书

本人声明所呈交的硕士学位论文是本人在导师指导下进行的研究工作及取得的研究成果。除了文中特别加以标注和致谢的地方外，论文中不包含其他人已经发表或撰写过的研究成果，也不包含为获得南京航空航天大学或其他教育机构的学位或证书而使用过的材料。

本人授权南京航空航天大学可以将学位论文的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或扫描等复制手段保存、汇编学位论文。

（保密的学位论文在解密后适用本承诺书）

作者签名：

日　　期：2021年3月27日

摘　要

在许多现实问题环境中，一个团队必须协调个体行为以实现整体利益的最大化，例如网络数据包路由、多机器人系统以及自动驾驶车辆的协调。多智能体强化学习为这样的协作问题提供了一种建模思路。本文围绕多智能体强化学习，分别从算法和应用层面进行了以下两方面研究。

第一，本文从算法层面研究协作型多智能体强化学习中的集中价值函数估计问题。目前，由于概念的简单性和实用性，大多数多智能体强化学习方法基于“集中训练、分散执行”的范式研究多智能体协作问题。为了做到更好的值函数估计，本文提出了一种基于“集中训练，分散执行”的多智能体强化学习算法——SMIX(λ)，该算法旨在学习一个稳定且具有可拓展性的集中价值函数。为实现这一点，本文所提方法充分结合了不同的因素，包括 1) 消除多智能体在学习阶段不切实际的集中贪婪假设，2) 使用 TD(λ) 构建多步回报实现价值函数估计时偏差和方差的权衡，3) 使用一种基于经验回放机制的异策略训练方法。在星际争霸游戏平台中的实验表明，本文所提方法在样本效率以及最终性能上均极大地超越了该平台中已有的最先进算法，同时本文所提出的集中价值函数估计方法可用于增强其他多智能体强化学习算法的价值函数，使其性能得到大幅度提升。

第二，本文以广告自动竞价问题为例，从多智能体系统的角度研究自动竞价智能体的设计。本文提出了一种多智能体强化学习框架，称为 MAAB (Multi-Agent Auto-bidding)，来学习自动竞价策略。首先，本文提出了一种温度调控的奖励分配机制来建立自动竞价智能体之间的混合协作竞争关系，从而实现广告主自身效用的最大化并且保证社会福利。其次，我们观察到协作关系会引导智能体走向共谋出低价的行为模式，从而损害平台的收益。为解决这个问题，我们引入了门槛智能体来为每一个自动竞价智能体设置一个个性化的竞价门槛，来降低平台收入的减少。在两智能体竞价环境中的实验表明，MAAB 在可以实现较高的社会福利的同时保证平台收入。

关键词：多智能体系统，强化学习，多智能体强化学习，多智能体协作，深度学习

ABSTRACT

In many real-world problems, such as network packet routing, and the coordination of multi-robot systems and autonomous vehicles, a team of units must coordinate their behaviors to achieve better social optimality. Multi-agent reinforcement learning (MARL) provides a possible solution for these collaborative problems. This thesis investigates deep multi-agent reinforcement learning from the perspectives of both algorithm and application.

First, we investigate the estimation of centralized value functions widely adopted in cooperative MARL. Recently the paradigm of centralized training with decentralized execution (CTDE) has become popular for multi-agent learning due to its conceptual simplicity and practical effectiveness. For a better estimation of the centralized value functions, we present a sample efficient and effective value-based method, named SMIX(λ). Our method carefully combines different elements, including 1) removing the unrealistic centralized greedy assumption during the learning phase, 2) using the λ-return to balance the trade-off between bias and variance, and 3) adopting an experience-replay style off-policy training. Experiments on the StarCraft Multi-Agent Challenge (SMAC) benchmark show that the proposed SMIX(λ) algorithm outperforms several state-of-the-art MARL methods by a large margin, and that it can be used as a general tool to improve the overall performance of a CTDE-type method by enhancing the evaluation quality of its centralized value function.

Second, we study the design of auto-bidding agents from the perspective of a distributed multi-agent system, and proposes a MARL framework for auto-bidding, namely MAAB, to learn the auto-bidding strategies. We first investigate the competition and cooperation relation among auto-bidding agents, and propose temperature-regularized credit assignment for establishing a mixed cooperative-competitive paradigm. By carefully making a competition and cooperation trade-off among the agents, we can reach an equilibrium state that not only guarantees individual advertiser's utility but also the system performance (social welfare). Besides, due to the observed collusion behaviors of bidding low prices underlying the cooperation, we further propose bar agents to set a personalized bidding bar for each agent, and then alleviate the degradation of revenue. Experiments on the two-agent bidding game demonstrate that our approach outperforms several baseline methods in terms of social welfare and guarantees the ad platform's revenue.

Keywords: Multi-Agent Systems, Reinforcement Learning, Multi-Agent Reinforcement Learning, Multi-Agent Cooperation, Deep Learning

目　录

图表清单

注释表

Q	Q 函数	V	V 函数
A	优势函数	$\mathbf{a}$	联合动作
s	状态	a	动作
$\mathcal{P}$	状态转移函数	γ	折扣因子
r	环境奖励	λ	TD(λ) 取值
ϵ	ϵ-贪心参数	τ	轨迹或温度参数
θ	策略参数	b	批大小
N	智能体个数	$\mathcal{A}$	动作空间
J	优化目标函数	$\mathcal{Z}$	Dec-POMDP 观测函数
$\mathbb{E}$	期望	$\mathcal{O}$	Dec-POMDP 观测空间
$\mathcal{S}$	状态空间	Q_{tot}	联合 Q 函数
B	剩余预算	v	展现价值
c	成本	ts	剩余展现机会

缩略词

缩略词	英文全称
CGB	Centralized Greedy Behaviour
CTDE	Centralized Training，Decentralized Execution
CVF	Centralized Value Functions
DDPG	Deep Deterministic Policy Gradient
DPG	Deterministic Policy Gradient
DL	Deep Learning
DNN	Deep Neural Networks
DQN	Deep Q-network
DRL	Deep Reinforcement Learning
IQL	Independent Q-Learning
IAC	Independent Actor-Critic
IS	Importance Sampling
MARL	Multi-Agent Reinforcement Learning
MC	Monte-Carlo
MCTS	Monte Carlo Tree Search
MDP	Markov Decision Process
NLP	Natural Language Processing
RL	Reinforcement Learning
RTS	Real-Time Strategy
SMAC	StarCraft Multi-Agent Challenge
VDN	Value Decomposition Networks

第一章 绪论

1.1 研究背景与意义

强化学习 (Reinforcement Learning, RL)[1] 是人工智能研究的一个重要分支，与监督学习 (Supervised Learning)[2]、无监督学习 (Unsupervised Learning)[3] 并称机器学习 (Machine Learning)[4] 三大学习范式。强化学习中的智能体主要通过与环境进行交互来进行学习，通过控制某个在环境中自主行动的个体 (或称智能体，agent)，与环境进行交互，通过感知、试探环境，由此获得环境的奖励或惩罚，从而不断改善自身决策能力。这种“试错学习” (trial-and-error) 同时具有主动适应和动态增强两个独特的优点，使其成为智能系统领域的核心技术之一。

然而，强化学习以及传统机器学习的挑战之一为难以手工设计可学习的高质量特征，这导致将强化学习应用到具有高维状态和动作空间的任务中较难，极大限制了强化学习的发展。近年来，深度学习 (Deep Learning, DL)[5] 由于其强大的特征表示能力受到广泛关注，并在计算机视觉[6–9]、语音识别[10–12]、自然语言处理[13–16] 等领域取得了显著的成功。受深度学习成功的启发，深度强化学习 (Deep Reinforcement Learning, DRL) 将深度学习的表示能力与强化学习的决策能力相结合，以提升强化学习解决高维复杂决策任务的能力。在深度强化学习中，深度神经网络被用以训练智能体策略以及值函数，通过将深度神经网络 (Deep Neural Networks, DNN) 作为函数近似器 (Function Approximator)，训练的策略及值函数往往具有较强的泛化能力。这大大解放了强化学习的能力，使得深度强化学习在机器人控制[17,18]、智能制造[19,20]、仿真模拟[21,22]、优化与调度[23,24] 以及广告竞价[25–27] 等领域中得到了广泛的应用。

深度多智能体强化学习[28] 作为深度强化学习的一个子分支，结合深度学习高维状态表示的优势，考虑在一个系统中同时有多个智能体与环境进行交互的问题。该模式下，每个智能体仍遵循着单智能体强化学习目标，即最大化期望回报，这种多智能体的建模方式是对单智能强化学习的拓展，同时也更符合对许多物理世界的建模。近年来，多智能体强化学习在多人游戏[29–31]，分布式控制[28]、资源分配[32,33]、作业调度[34] 以及实时竞价[35] 等领域的应用越来越广泛。例如著名的围棋程序 Alpha Go[29] 解决的是一个典型的二人博弈多智能体强化学习问题，OpenAI 研发的 OpenAI Five[31] 以及 DeepMind 研发的 AlphaStar[30] 则是解决多智能体强化学习的协同问题。随着强化学习的数学理论以及博弈论的研究取得进展，深度多智能体强化学习受到了国内外学者的广泛关注，并成为人工智能领域的一个研究热点，目前深度多智能体强化学习已发展成为机器学习领域最富有挑战性和最活跃的子领域之一。

(a) OpenAI Five　　(b) AlphaStar

图 1.1　OpenAI 与 DeepMind 分别研发了基于多智能体强化学习的游戏 AI，在 Dota2 和星际争霸 2 游戏中均达到了人类级别的水平。

1.2　国内外研究现状

近年来，由于深度神经网络强大的表示能力，深度强化学习取得了显著的进展[29,36–38]。然而，多智能体场景下依然具有较多单智能体强化学习中所不具备的挑战，这些挑战使得很难直接将深度强化学习应用到多智能体领域中去[39–42]。主要的挑战包含以下几点：

- 联合动作随智能体数量指数增长[41,43]。如果训练一个集中策略，那么该策略的输出动作空间 $|A|$ 将随智能体数量 N 呈指数增长，也就是说，N 个智能体的联合动作空间为 $|A|^N$，联合动作空间的指数爆炸问题使得集中式的方法难以直接应用到多智能体问题中。
- 环境的非静态特性[44]。在一个多智能体系统中，个体在学习过程中将其他个体视为环境的一部分，然而，从任何给定个体的角度来看，环境中多个学习个体的存在使得学习问题是非静态的，即给定个体的状态动作元组 (s, a)，该个体的期望回报依赖于该系统中其他个体所部署的策略。然而，所有个体在学习过程中在不断改变其策略和行为，这导致个体的期望回报也是不断变化的。
- 奖励分配 (Credit Assignment) 问题[40]。在协作场景中，环境针对联合动作通常以联合奖励的形式予以反馈，这使得每个智能体很难推断出自己对整体奖励的具体贡献是多少。尽管在某些情况下，可以为每个智能体设计个体奖励以减少这个问题的严重性，然而这些奖励在合作场景中一般难以获得，而且往往无法鼓励个体牺牲自身利益以获取更大的联合利益。

解决联合动作空间“维度诅咒”问题的最简单方法为独立学习[45]，即直接使用单智能体算法为每一个智能体学习一个策略。由于该方法并不具有一个集中式的策略，因此不受上述联合动作指数爆炸的影响。然而该方法受到环境非静态因素的影响，导致基于独立学习的策略常常无法收敛[39]。此外，对于多智能体协作问题，该方法缺少一种奖励分配机制，使得智能体行为无法受到适当的奖励信号的引导。但由于独立学习概念上的简单性，该方法通常作为多智能体

强化学习中的基准算法。

目前，基于“集中训练、分散执行”(Centralized Training, Decentralized Execution, CTDE) 的学习模式成为多智能体强化学习领域的一个新的学习范式。该范式将智能体训练和执行阶段剥离开，通过在训练阶段引入更多的全局信息[41] 以减少环境的非静态特性对智能体学习所造成的影响，该范式只要求在训练阶段引入的额外信息在部署执行阶段无需依赖即可。因此可以通过在训练阶段引入其他智能体的信息以减少环境的非静态性所造成的影响。这种范式目前成为了解决 Dec-POMDP[46] 框架下的问题的一种标准范式。目前基于“集中训练，分散执行”范式的代表方法有 COMA[40], MADDPG[47], VDN[48], QMIX[41] 以及 QTRAN[49]。下面进行简要介绍。

COMA[50] 是一种基于 Actor-Critic[51] 结构的多智能体强化学习算法，旨在解决多智能体强化学习中的奖励分配问题[52]。在合作场景中，环境对于联合动作通常仅反馈以联合奖励，这使得每个智能体很难推断自己对联合奖励的贡献。因此 COMA 在这种设定下，基于差分奖励[53] 的思想，使用集中基于反事实基准 (counterfactual baseline) 来计算每个策略的优势函数。

MADDPG[47] 则是一种基于 DDPG[54] 的多智能体强化学习算法。该算法基于 CTDE 范式，在训练阶段为每一个智能体的策略分别训练一个集中价值函数，通过在集中价值函数的训练过程中引入额外信息来缓解环境的非静态问题。此外, 与 COMA 不同，MADDPG 可用于竞争、协作以及混合竞争与协作条件下，只需调整不同智能体奖励函数之间的关系，即可改变智能体之间的关系为竞争、协作或者介于两者中间的一种状态。

与 COMA 和 MADDPG 不同，VDN、QMIX 以及 QTRAN 属于值分解一类的多智能体方法。值分解网络 (Value Decomposition Networks, VDN)[48] 旨在学习一个联合 Q_{tot} 函数，使用该联合 Q_{tot} 函数对所有智能体的联合动作进行状态动作值函数评估。QMIX[41] 将 VDN 的加性值分解扩展成非线性分解，通过在集中价值函数结构上施加的约束来保证非线性单调得以保证。基于 QMIX 所存在的结构约束，QTRAN[49] 提出学习一个集中式无约束的联合 Q 函数以及一个 VDN 分解的联合 Q 函数，由于无约束的 Q 值无法被有效最大化，VDN 分解的 Q 值被用来产生近似最大的联合动作，这使得 QTRAN 能够表示比 QMIX 假设空间更大的联合 Q 函数。

1.3 本文主要研究工作

本文主要从算法和应用两方面关注基于深度强化学习下的多智能体协作问题。

考虑在一个复杂动态场景中，多个智能体通过与环境协同探索和行动，以完成团体共同的目标。在算法层面，目前大多数方法，例如 COMA[50], MADDPG[47], QMIX[41] 以及 QTRAN[49] 均基于 CTDE 的范式研究多智能体协作问题，但集中价值函数在 CTDE 范式中所发挥的核心作用在当前领域中尚未得到足够的重视[40,41,47,48]。多数工作通常用单智能体设定下的方式来学习集中价值函数，这导致在多智能体环境中会引入估计误差以及较大的方差。另一方面，由于以下原因，估计多智能体环境中的集中价值函数是困难的: 1) 多智能体联合动作空间的“维度诅

咒” 导致经验稀疏[55]；2) 环境非马尔可夫性质的挑战以及多智能体环境中的部分可观测性比在单智能体环境更为严峻；3) 多智能体环境的复杂且难以建模，部分原因是智能体之间的交互复杂。这些因素通常会导致集中价值函数的估计不稳定，且偏差和方差很高。

为了解决这些困难，本文提出了一个新的具有较高样本效率，基于 CTDE 框架的多智能体强化学习方法，称为 SMIX(λ)。SMIX(λ) 通过基于异策略的集中价值函数学习方法改进了集中价值函数估计，该方法消除了在训练过程中对集中贪婪行为 (Centralized Greedy Behaviour, CGB) 假设的需要，并且引入的 λ-回报[1] 可以更好地平衡偏差和方差。SMIX(λ) 使用异策略学习机制由重要性采样 (Importance Sampling, IS)[56] 驱动的，但是通过经验回放 (Experience Replay) 机制[36] 来实现，这种机制与先前的单智能体学习中的 Q(λ)[57] 方法有着密切的联系。通过结合这些要素，SMIX(λ) 方法有效地提高了采样效率并稳定了训练过程。本文以星际争霸多智能体挑战赛 (StarCraft Multi-Agent Challenge, SMAC)[58] 为基准进行实验，SMIX(λ) 几乎可以在所有的场景中获得了最好的性能。此外，通过将现有的 CTDE 型多智能体强化学习算法的集中价值函数估计器替换为本文提出的 SMIX(λ)，可以发现现有 CTDE 型多智能体强化学习算法可以获得显著的性能提升。

在应用层面，本文以广告自动竞价问题为例[59]，考虑从多智能体系统的角度来对该问题进行建模。在自动竞价问题中，广告平台为广告主设计自动出价智能体，每个智能体都为每一类具有相同目标的广告主学习一种特定的出价策略，以在一定约束下优化广告主期望的目标。例如，在一般的广告平台中，存在三种典型的自动出价智能体：CLICK 智能体，CONV 智能体和 CART 智能体，它们分别在预算约束下优化点击次数，购买转化次数以及添加到购物车的数量。这些自动出价智能体通过竞标每个广告展现机会来相互竞争。在每一次广告展现机会到来后，平台将为参与的自动出价智能体发起广告拍卖，每次拍卖中，自动竞价智能体都根据广告主的目标和约束以及预测值（例如，pCTR，pCVR 或 pCART）为这个展现机会出价。在收到所有智能体的出价之后，拍卖机制决定获胜的广告主和相应的展现成本。

本文针对自动竞价问题提出了一种混合协作竞争的多智能体强化学习框架，称为 MAAB。首先，为了实现自动竞价智能体之间协作与竞争关系的折中，本文提出了一种基于温度调控的奖励分配方法，通过一种基于 softmax 的方式将总体奖励按与智能体出价正比的方式进行分配，总而建立了自动竞价智能体之间混合协作竞争的关系。同时，softmax 函数中的温度调控参数可以作为一种用以调控智能体之间协作性与竞争性的工具。其次，受最优拍卖中[60–62] 的保留价启发，本文设计门槛智能体来为每一个自动竞价智能体学习一个具有个性化的竞价门槛，以减少平台收入的降低。直觉上，门槛智能体的目标是提高竞价门槛以提高平台的收入，然而自动竞价智能体则具有一个相反的目标，即降低竞价门槛使得可以以较低的价格赢得广告展现机会。门槛智能体是以一种对抗的方式与自动竞价智能体进行联合训练，直到双方的策略达到某种均衡点。本文在两个智能体的自动竞价环境中对本文所提方法进行验证，实验表明本文方法

可以在提高社会福利的同时，减少平台收入的降低。

1.4 本文的内容安排

本文内容共分为五章, 具体的结构安排如下:

第一章，绪论。首先介绍了选择将多智能体强化学习作为课题的研究背景和意义，并简要概括了目前国际上针对该问题的一些挑战以及研究现状。此外, 交代了本文的主要研究工作, 最后给出全文的结构安排。

第二章，首先介绍马尔可夫决策过程 (Markov Decision Process, MDP) 的基本概念，同时介绍策略评估和策略迭代的相关概念，这些概念是后续所介绍的强化学习算法的基础。这些算法包括策略梯度 (Policy Gradient, PG)[63]，SARSA[1]，期望 SARSA[1]，Q-learning[64] 以及结合深度神经网络的深度 Q 网络 (Deep Q-networks, DQN)[36] 和 DDPG (Deep Deterministic Policy Gradient)[54]。

第三章，介绍一些经典的多智能体强化学习协作算法，这部分介绍的算法也作为第四章实验部分的对比算法。首先介绍基于独立学习[45] 的两个方法——独立 Actor-Critic (Independent Actor-Critic, IAC) 和独立 Q 学习 (Independent Q-learning, IQL)。然后基于独立学习所存在问题，进一步介绍基于 Actor-Critic 的 COMA[50] 和基于 DDPG[54] 的 MADDPG 算法[47]。最后介绍基于值分解方法的 VDN[48] 和 QMIX[41]。

第四章，介绍本文的第一个主要工作，这部分提出了一种增强多智能体集中价值函数估计的一种强化学习协作算法。这部分首先对多智能体集中价值函数估计的挑战进行了说明, 通过对集中价值函数的假设空间加以分析，进而提出本文的算法——SMIX(λ)，最后通过充分的实验比较了本文算法相对已有算法的优越性及本文方法在集中价值函数估计中的通用性。

第五章，介绍本文的第二个主要工作，这部分以广告自动竞价问题为例，提出了一种多智能体强化学习框架 MAAB，用以解决自动竞价策略设计的问题。这部分首先对自动竞价问题进行了简单的背景介绍, 然后提出了该部分的主要框架 MAAB，最后通过实验比较了 MAAB 与一些基准算法的性能，并通过消融实验分析了 MAAB 中门槛智能体在自适应设置竞价门槛上的有效性。

第六章，总结与展望。总结了本文所有的研究工作，并指出了本文的方法可继续改进的地方，此外展望了未来的研究方向。

第二章　MDP 与强化学习

本节中，首先介绍强化学习一般用以建模环境的框架——马尔可夫决策过程 (MDP)，然后介绍策略评估与策略迭代框架，最后介绍强化学习的相关基础算法，具体包括 SARSA[1]，期望 SARSA[1]，Q-learning[1]、深度 Q 网络[36] 以及 DDPG[54]，这些算法是下一章节多智能体强化学习算法的基础。

2.1　马尔可夫决策过程

在强化学习中，每个智能体通过与环境进行交互以最大化其累积回报 (见图2.1)。环境一般通过马尔可夫决策过程 (MDP) 进行建模。具体地说，MDP 是一个五元组 $(\mathcal{S}, \mathcal{A}, P, R, \gamma)$，每个智能体将观测到的环境相关信息作为其状态 $s \in S$，然后基于状态 s，根据其策略 π 采取其相应的动作 $a = \pi(s)$，然后环境对智能体的行为以奖励 $r(s, a)$ 的形式予以反馈，并且根据状态转移函数 $P(s'|s, a)$ 转移到下一状态 s'。γ 为奖励的折扣因子，表达了长远奖励随时间的衰减率。在 MDP 中，一种标准的假设为智能体无法获知环境的状态转移函数以及奖励函数，智能体通过与环境的交互过程感知这些信息。

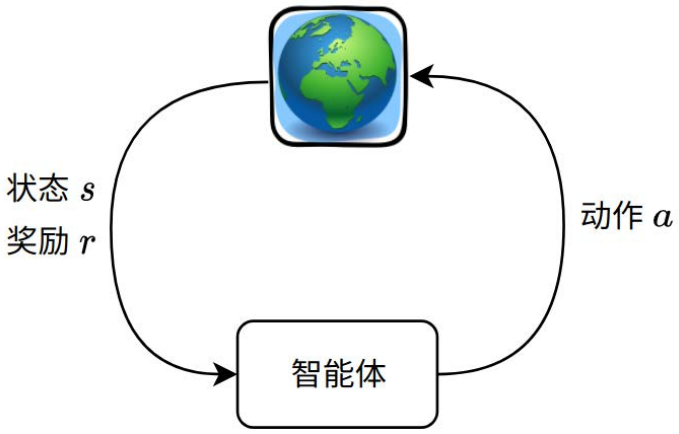


图 2.1　智能体与环境交互示意图。智能体获得环境的状态 s 选择采取动作 a，环境基于智能体采取的动作反馈奖励 r 以及下一状态，于是智能体根据一下状态继续采取动作，如此反复……

在 MDP 框架下，智能体的目标为学得最优策略 π^*, 使得该策略可以最大化回报，即

$$\max_{\pi} \mathbb{E}_{\tau\sim\pi}\left[R_0(\tau)\right], \tag{2.1}$$

其中，$R_t(\tau) = \sum_{t'=t} \gamma^{t'-t} r_{t'}$ 为从时刻 t 开始的回报，$\tau = \{s_0, a_0, r_0, \cdots, s_{T+1}\}$ 为使用策略 π 采样的轨迹，基于马尔可夫性质，其概率通常可表达为 $\pi(\tau) = P(s_0)\prod_{t=0}^{T} P(s_{t+1}|s_t, a_t)\pi(a_{t+1}|s_{t+1})$。

可定义相应策略 π 的价值函数 (Value Functions)，价值函数一般用以对策略进行评估，策略的价值函数表达了该策略在不同状态下的期望长远回报。常见的值函数有 $V^\pi(s)$ 函数和 $Q^\pi(s,a)$ 函数，分别定义了策略 π 在状态 s 下的期望长远回报以及在状态 s 下采取动作 a 的期望长远回报，详细定义如下：

$$V^\pi(s_t) = \mathbb{E}_{a_t, s_{t+1}, \cdots}\left[R_t(\tau)\right] \tag{2.2}$$

$$Q^\pi(s_t, a_t) = \mathbb{E}_{s_{t+1}, a_{t+1}, \cdots}\left[R_t(\tau)\right], \tag{2.3}$$

其中 $a_t \sim \pi(a_t|s_t)$, $s_{t+1} \sim P(s_{t+1}|s_t, a_t), \forall t \geq 0$。此外，一般还可定义优势函数 (advantage function)[65]:

$$A^\pi(s,a) = Q^\pi(s,a) - V^\pi(s). \tag{2.4}$$

直觉来看，优势函数表达了在状态 s 采取动作 a 所得到的期望长远回报相对于平均动作的优势，因此其期望 $\mathbb{E}_{a\sim\pi}\left[A^\pi(s,a)\right] = 0$。

2.2 策略评估与迭代

策略评估考虑对一个已有策略进行性能评估，是对策略进行改进的基础。而策略迭代包括对策略的评估和策略的提升，构成了策略学习的整体框架。本节分别对策略评估和策略迭代进行简要介绍。

2.2.1 策略评估

在动态规划中，策略评估一般指给定任意策略 π，计算其价值函数 V_π。注意到状态值函数 $V_\pi(s) = \mathbb{E}[R_t|s_t = s]$ 满足以下的递归性质：

$$\begin{aligned} V_\pi(s) &\doteq \mathbb{E}[G_t|s_t = s] \\ &= \mathbb{E}_\pi[r_{t+1} + \gamma R_{t+1}|s_t = s] \\ &= \sum_a \pi(a|s) \sum_{s',r} P(s', r|s, a)[r + \gamma V_\pi(s')]. \end{aligned} \tag{2.5}$$

式 (2.5) 也称 V_π 的贝尔曼方程 (Bellman Equation), 由式 (2.5) 可见，策略评估算法通过迭代地使用贝尔曼方程作为更新规则来计算当前状态的价值函数 $V_\pi(s)$。策略评估具体流程见算法1。

2.2.2 策略改进

计算策略价值函数的目的是为了找到一个更好的策略。在上述策略评估的基础上，策略改进考虑对一个非最优策略进行性能提升，即按照某种方式修改策略 π 以得到策略 π'，使得 $V_{\pi'}(s) \geq V_\pi(s), \forall s \in S$。对于所有的状态-动作对, 如果一个策略的期望回报大于或等于其他所有策略的期

Algorithm 1: 迭代策略评估

Input: 需要评估的策略 π

初始化数组 $V(s) = 0, \forall s \in \mathcal{S}$, ϵ 为一个很小的正数，决定了评估的精度;

while *True* **do**

$\Delta \leftarrow 0$;

for 每一个状态 $s \in \mathcal{S}$ **do**

$v \leftarrow V(s)$;

$V(s) \leftarrow \sum_a \pi(a|s) \sum_{s',r} P(s', r|s, a)[r + \gamma V_\pi(s')]$;

$\Delta \leftarrow \max(\Delta, |v - V(s)|$;

if $\Delta < \epsilon$ **then**

break;

望回报, 那么称该策略为最优策略。策略改进的最终目的就是得到最优策略。最优策略可能有多个, 但它们共享一个状态值函数:

$$V_*(s) \doteq \max_\pi V_\pi(s), \quad \forall s \in \mathcal{S}. \tag{2.6}$$

同样也共享一个状态-动作值函数:

$$Q_*(s, a) \doteq \max_\pi Q_\pi(s, a), \quad \forall s \in \mathcal{S}, a \in \mathcal{A}. \tag{2.7}$$

通过观察式 (2.6) 和式 (2.7)，可以发现状态值函数和状态动作值函数具有下面的关系:

$$\begin{aligned} v_*(s) &\doteq \max_\pi V_\pi(s) \\ &= \max_{a \in \mathcal{A}(s)} Q_*(s, a). \end{aligned} \tag{2.8}$$

因此, 对于当前策略 π，策略评估一般通过将当前策略选择的动作改变为当前最优的动作来对策略进行提升，即:

$$\begin{aligned} \pi'(s) &\doteq \arg\max_a Q_\pi(s, a) \\ &= \arg\max_a \mathbb{E}[r_{t+1} + \gamma V_\pi(s_{t+1})|s_t = s, a_t = a] \\ &= \arg\max_a \sum_{s',r} P(s', r|s, a)[r + \gamma V_\pi(s')]. \end{aligned} \tag{2.9}$$

2.2.3 策略迭代与值迭代

有了策略改进的方法, 那么获得最优策略的过程便为: 从一个初始策略 (通常是随机策略) 出发, 先进行策略评估, 然后改进策略, 评估改进的策略, 再进一步改进策略……不断迭代进行策略

评估和改进, 直到策略不再改变为止。这样获取最优策略的方法称策略迭代 (policy iteration)[1], 策略迭代的具体流程见算法2。

Algorithm 2: 策略迭代

1. 初始化

$V(s) \in \mathbb{R}, \pi(s) \in \mathcal{A} \quad \forall s \in \mathcal{S}, \epsilon$ 为一个较小的正数，决定策略评估的精度;

2. 策略评估

while *True* **do**

  $\Delta \leftarrow 0$;

  for 每一个状态 $s \in \mathcal{S}$ **do**

    $v \leftarrow V(s)$;

    $V(s) \leftarrow \sum_a \pi(a|s) \sum_{s',r} P(s', r|s, a)[r + \gamma V_\pi(s')]$;

    $\Delta \leftarrow \max(\Delta, |v - V(s)|$;

  if $\Delta < \epsilon$ **then**

    break;

3. 策略改进

`policy_stable` $\leftarrow$ true;

for 每一个状态 s **do**

  `old_action` $\leftarrow \pi(s)$;

  $\pi(s) \leftarrow \arg\max_a \sum_{s',r} P(s', r|s, a)[r + \gamma V(s')]$;

  if *old_action* $\neq \pi(s)$ **then**

    `policy_stable` $\leftarrow$ false;

if *policy_stable* **then**

  停止并返回 $V \approx v_*$ 和 $\pi \approx \pi_*$;

else

  返回步骤 2;

策略迭代的一个缺点是其每一次迭代都包含了策略评估过程，该过程通常比较耗时，因此可以采用值迭代 (value iteration)[1]，值迭代的具体流程见算法3。

策略和值迭代的框架可以让我们基于已知的环境模型学习到最优策略，但上述的讨论依然基于环境模型已知的情况下 (环境的状态转移概率和奖励函数均已知)。然而在实际情况中，环境的模型往往是未知的，下面一节介绍基于无模型 (model-free) 的强化学习算法，即在环境模型未知的情况下进行策略学习。

Algorithm 3: 值迭代

初始化数组 $V(s)$(例如 $V(s) = 0,\ \forall s \in \mathcal{S}$),$\epsilon$ 为一个较小的正数，决定策略评估的精度;

while *True* **do**

  $\Delta \leftarrow 0$;

  for 每一个 $s \in \mathcal{S}$ **do**

    $v \leftarrow V(s)$;

    $V(s) \leftarrow \max_a \sum_{s',r} P(s', r|s, a)[r + \gamma V_\pi(s')]$;

    $\Delta \leftarrow \max(\Delta, |v - V(s)|)$;

  if $\Delta < \epsilon$ **then**

    break;

输出一个确定性策略 $\pi \approx \pi_*$ 使得 $\pi(s) = \arg\max_a \sum_{s',r} P(s', r|s, a)[r + \gamma V_\pi(s')]$;

2.3 强化学习算法介绍

本节中介绍几种比较重要的单智能体强化学习算法，具体包括 SARSA，期望 SARSA，Q-learning 以及深度 Q 网络 (Deep Q-network, DQN)。这些算法也是本文实验部分所对比的多智能体强化学习算法的基础算法。

2.3.1 策略梯度方法

策略梯度方法首先定义一个评价策略性能的指标，然后通过计算梯度来对策略进行更新。首先约定 θ 为策略的参数向量, $\pi_\theta(a|s)$ 为智能体的策略。策略梯度方法的步骤大致为: 1) 首先定义评估策略表现的代价函数 $J(\theta)$, 2) 求解 $J(\theta)$ 关于参数 θ 的偏导数, 3) 通过梯度上升 (gradient ascent) 更新参数 θ。下面简要介绍进行基于蒙特卡洛采样方法的策略梯度推导。

使用策略 π_θ 产生一条轨迹 $\tau =< s_1, a_1, \cdots, s_T, a_T >$, 产生该条轨迹的概率可以表达为:

$$\begin{aligned} \pi_{\boldsymbol{\theta}}(\tau) &= p_{\boldsymbol{\theta}}\left(s_1, a_1, \ldots, s_T, a_T\right) \\ &= p\left(s_1\right) \prod_{t=1}^{T} \pi_{\boldsymbol{\theta}}\left(a_t \mid s_t\right) p\left(s_{t+1} \mid s_t, a_t\right). \end{aligned} \tag{2.10}$$

为评估策略性能, 首先定义一个评估策略性能的代价函数 $J(\theta)$, 可通过期望累积奖赏之和作为评估该策略好坏的标准，然而在实际的强化问题中期望累积奖赏难以获得, 因此可通过策略 π_θ 产

生多条轨迹, 使用多条轨迹的累积奖赏均值作为期望累积奖赏的近似，即：

$$J(\boldsymbol{\theta}) = \mathbb{E}_{\tau \sim p_{\boldsymbol{\theta}}(\tau)} \left[\sum_{t=1}^{T} r\left(s_t, a_t\right) \right] \tag{2.11}$$

$$\approx \frac{1}{n} \sum_{i=1}^{n} \sum_{t=1}^{T} r\left(s_{i,t}, a_{i,t}\right), \tag{2.12}$$

其中 n 为轨迹数目。为求得最优参数 θ^*, 需要先求梯度, 然后迭代更新参数 θ 直至稳定, 梯度 $\nabla J(\theta)$ 为:

$$\begin{aligned} \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) &= \int \nabla_{\boldsymbol{\theta}} \pi_{\boldsymbol{\theta}}(\tau) r(\tau) d\tau \\ &= \int \pi_{\boldsymbol{\theta}}(\tau) \frac{\nabla_{\boldsymbol{\theta}} \pi_{\boldsymbol{\theta}}(\tau)}{\pi_{\boldsymbol{\theta}}(\tau)} r(\tau) d\tau \\ &= \int \pi_{\boldsymbol{\theta}}(\tau) \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(\tau) r(\tau) d\tau \\ &= \mathbb{E}_{\tau \sim \pi_{\theta}(\tau)} \left[\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(\tau) r(\tau) \right]. \end{aligned} \tag{2.13}$$

式 (2.13) 中 $\log \pi_{\theta}(\tau)$ 的可以利用通过对式 (2.10) 两边分别取对数获得。通过采样 n 条轨迹，策略梯度可以使用采样的 n 条轨迹近似为：

$$\begin{aligned} \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) &= \mathbb{E}_{\tau \sim \pi_{\boldsymbol{\theta}}(\tau)} \left[\left(\sum_{t=1}^{T} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}\left(a_t \mid s_t\right) \right) \left(\sum_{t=1}^{T} r\left(s_t, a_t\right) \right) \right] \\ &\approx \frac{1}{n} \sum_{i=1}^{n} \left[\left(\sum_{t=1}^{T} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}\left(a_{i,t} \mid s_{i,t}\right) \right) \left(\sum_{t=1}^{T} r\left(s_{i,t}, a_{i,t}\right) \right) \right] \\ &= \frac{1}{n} \sum_{i=1}^{n} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}\left(\tau_i\right) r\left(\tau_i\right). \end{aligned} \tag{2.14}$$

上述即基于蒙特卡洛采样的策略梯度的基本思想，该算法一般也称 REINFORCE，算法4给出了 REINFORCE 算法的流程。

Algorithm 4: REINFORCE 算法

Result: 输出策略参数 θ

while *True* **do**

从策略 $\pi_{\theta}\left(a_t \mid s_t\right)$ 中采样轨迹 $\left\{\tau^i\right\}$;

计算 $J(\boldsymbol{\theta}) \approx \frac{1}{n} \sum_{i=1}^{n} \left[\left(\sum_{t=1}^{T} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}\left(a_{i,t} | s_{i,t}\right) \right) \left(\sum_{t=1}^{T} r\left(s_{i,t}, a_{i,t}\right) \right) \right]$;

$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$;

end

尽管上述基于蒙特卡洛采样的策略梯度方法是无偏的，但其通常具有较大的方差，并且收敛速度较慢。为了降低策略梯度的方差，一种可行的方法是引入状态动作值函数作为 critic，并作为策略性能的评估度量。

式 (2.14) 中推导策略梯度可展开为:

$$\begin{aligned}\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) &= \frac{1}{|D|}\sum_{\tau\in D}\Big(\sum_{t=0}^{T}\nabla_{\boldsymbol{\theta}}\log\pi_{\boldsymbol{\theta}}(a_t|s_t)\Big)R(\tau)\\ &= \frac{1}{N}\sum_{i=1}^{N}\sum_{t=0}^{T}\nabla_{\boldsymbol{\theta}}\log\pi_{\boldsymbol{\theta}}(a_{i,t}|s_{i,t})\Big(\sum_{t'=0}^{T} r(s_{i,t'},a_{i,t'})\Big),\end{aligned} \tag{2.15}$$

其中 $\tau=(s_0,a_0,s_1,\dots)$ 为一条轨迹，D 是通过策略 $\pi_{\boldsymbol{\theta}}$ 采样产生的轨迹的集合，$R(\tau)=\sum_{t=0}^{T}\gamma^t r_t$ 为一条轨迹的折扣累积奖赏。

为解决基于轨迹的策略梯度方差较大的问题，可以采用采用基于状态-动作的策略梯度计算方式以减小方差。这种方式计算的策略梯度为:

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T}\nabla_{\boldsymbol{\theta}}\log\pi_{\boldsymbol{\theta}}(a_{i,t}|s_{i,t})\Big(\underbrace{\sum_{t'=t}^{T} r(s_{i,t'},a_{i,t'})}_{Q(s_{i,t'},a_{i,t'})}\Big). \tag{2.16}$$

有时又将 $\sum_{t'=t}^{T} r(s_{i,t'},a_{i,t'})$ 称作 “reward to go", 这种累积奖赏的形式非常类似于状态-动作值函数 $Q_\pi(s_t,a_t)$，因此 Actor-Critic 通过额外学习一个策略的动作函数 $Q_\pi(s,a)$ 替换上式中的 “reward to go"，即:

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T}\nabla_{\boldsymbol{\theta}}\log\pi_{\boldsymbol{\theta}}(a_{i,t}|s_{i,t})Q_\pi(s_{i,t},a_{i,t}). \tag{2.17}$$

与 REINFORCE 相比，Actor-Critic 方法一般可以有效降低策略梯度的方差。

除了使用动作值函数替换 “reward to go" 之外，还可以将策略梯度表示为一种更加通用的形式:

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T}\nabla_{\boldsymbol{\theta}}\log\pi_{\boldsymbol{\theta}}(a_{i,t}|s_{i,t})\Phi_{i,t}, \tag{2.18}$$

其中 $\Phi_{i,t}$ 可以是以下几种形式[66]:

1. $Q_\pi(s_{i,t},a_{i,t})$: 动作值函数;
2. $A^\pi(s_{i,t},a_{i,t})$: 优势函数;
3. $\sum_{t'=t}^{T} r(s_{i,t'},a_{i,t'})$: “reward to go'';
4. $\sum_{t'=t}^{T} r(s_{i,t'},a_{i,t'})-b(s_{i,t})$: “reward to go" 减去一个基准的版本;
5. $r_{i,t}+V_\pi(s_{i,t+1})-V_\pi(s_{i,t})$: TD 冗余项。

2.3.2 SARSA

SARSA[67] 是基于策略迭代框架的一种强化学习方法，其名称 SARSA 是 “state–action–reward–state–action” 中每个单词首字母的缩写，表明该算法使用 (s,a,r,s',a') 进行动作值函数的更新。SARSA 的更新方式如下:

$$Q(s,a)\leftarrow Q(s,a)+\alpha[r+\gamma Q(s',a')-Q(s,a)], \tag{2.19}$$

其中 α 是学习率。动作 a 一般通过 ϵ-greedy 方式实现，即在状态 s 以概率 ϵ 选择随机动作，以确保智能体可以探索未知的环境，而以 $1-\epsilon$ 的概率选取可以令 Q 值最大的动作，即 $a = \arg\max_a Q(s,a)$。后述该类基于值的算法均可采取 ϵ-greedy 选取动作。

由于在下一状态 s'，SARSA 采取的动作为当前策略输出的动作 a'，即用于策略学习的数据均由当前策略产生，因此 SARSA 也被称为同策略 (on-policy) 学习算法。

2.3.3 期望 SARSA

在 SARSA[67] 的基础上，期望 SARSA[1] (expected SARSA) 在选取下一动作 a' 时使用期望形式，而非 SARSA 使用的对 a' 进行采样方式计算。期望 SARSA 的动作值函数更新方式如下：

$$Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma \mathbb{E}_\pi[Q(s',a')] - Q(s,a)] \tag{2.20}$$

$$= Q(s,a) + \alpha[r + \gamma \sum_{a'} \pi(a'|s')Q(s',a') - Q(s,a)]. \tag{2.21}$$

尽管期望 SARSA 在计算上比 SARSA 更复杂 (期望 SARSA 需计算下一状态下所有动作的概率)，但该算法减少了使用策略 π 随机采样 a' 所造成的误差，在目标的估计上一般来说更为准确。

2.3.4 Q-learning

前述 SARSA 与期望 SARSA 均为同策略算法，即 a' 的选取均取决于当前策略 π。而 Q-learning 使用一种异策略 (off-policy) 的方式选取下一动作 a'，其动作值函数的更新规则如下：

$$Q^*(s,a) \leftarrow Q^*(s,a) + \alpha\left[r + \gamma \max_{a'} Q^*(s',a')\right]. \tag{2.22}$$

尽管在公式上与 SARSA 差异不大，但 Q-learning 基于贝尔曼最优方程直接优化最优策略 π^*，其优化的动作值函数与所遵循的策略无关。而 SARSA 则为基于策略评估-策略提升框架下的算法，其优化的为当前策略的动作值函数。

2.3.5 深度 Q 网络

深度 Q 网络 (Deep Q-network, DQN)[36] 使用深度神经网络来参数化上述动作值函数 $Q(s,a)$，但使用神经网络无法保证 Q-learning 收敛，为此，深度 Q 网络[36] 提出使用经验回放机制稳定学习过程。经验回放机制维护一个缓冲池，其中存放了过去的经验，具体可描述为元组 (s,a,r,s')，其中 s' 为智能体在状态 s 采取动作 a 得到奖励 r 并转移到的下一个状态。神经网络参数 θ 通过最小化下列平方误差进行更新：

$$\mathcal{L}(\theta) = \sum_{i=1}^{b} (y_i^{DQN} - Q_\theta(s,a))^2. \tag{2.23}$$

其中 $y_i^{DQN} = r + \gamma \max_{a'} Q_{\theta_{\text{targ}}}(s', a')$，$\theta_{\text{targ}}$ 为目标网络 (target network) 参数，通过每隔固定时间间隔将当前 $Q_\theta(s, a)$ 的参数拷贝到 $Q_{\theta_{\text{targ}}}(s, a)$ 得到，算法5给出了 DQN 算法的训练流程。

通过使用经验回放机制以及目标网络，DQN 在 Atari 2600 游戏上达到了人类级别的性能，正式掀起深度强化学习的研究热潮。

Algorithm 5: DQN 算法流程

Input: 初始化 Q 函数参数 θ，空的经验缓冲池 $\mathcal{D}$

设置目标网络参数为主网络参数 $\theta_{\text{targ}} \leftarrow \theta$;

for 每一条轨迹 **do**

  for 轨迹的每一个时刻 **do**

    以概率 ϵ 随机选取动作 a，否则选取 $a = \arg\max_{a'} Q_\theta(s', a')$;

    在环境中执行动作 a，获得奖励 r 以及下一状态 s';

    将 (s, a, r, s') 存入 $\mathcal{D}$ 中;

    从 $\mathcal{D}$ 中采样批量数据 $B = \{(s, a, r, s')\}$;

    计算 TD 目标：

$$y(r, s', d) = r + \gamma(1 - d) \max_{a'} Q_{\theta_{\text{targ}}}(s', a';)$$

    通过梯度下降更新 Q 函数：

$$\nabla_\theta \frac{1}{|B|} \sum_{(s,a,r,s',d) \in B} \left(Q_\theta(s, a) - y\left(r, s', d\right)\right)^2$$

    每个 C 步更新目标网络：$\theta_{\text{targ}} \leftarrow \theta$;

2.3.6 DDPG

DDPG (Deep Deterministic Policy Gradient)[54] 通过将深度神经网络与 DPG (Deterministic Policy Gradient)[68] 算法相结合，构造出解决连续动作空间下序列决策问题的强化学习算法。主要动机在于 DQN 只能够解决低维离散动作空间的序列决策问题，然而许多问题则具有连续动作空间属性，例如在机器人控制中，机器人的动作一般定义为角度偏移量或者位移量，这些量一般为连续值。因此 DDPG 将 DQN[36] 拓展到连续动作空间中。

与 DQN 类似，DDPG 也直接学习最优动作值函数 Q^*，贝尔曼最优方程为：

$$Q^*(s, a) = \mathop{\mathrm{E}}_{s' \sim P}\left[r(s, a) + \gamma \max_{a'} Q^*\left(s', a'\right)\right]. \tag{2.24}$$

根据上述最优贝尔曼方程，可以构造出学习最优动作值函数的平方差损失：

$$L(\phi, \mathcal{D}) = \mathop{\mathrm{E}}_{(s,a,r,s',d) \sim \mathcal{D}}\left[\left(Q(s, a; \phi) - \left(r + \gamma \max_{a'} Q_{\phi_{\text{targ}}}\left(s', a'\right)\right)\right)^2\right], \tag{2.25}$$

其中 $Q(s,a;\phi)$ 是参数化的动作值函数，与 DQN 类似，DDPG 也是用了 $Q_{\phi_{targ}}(s,a)$ 作为延迟的目标网络以稳定训练，$\mathcal{D}$ 是类似上述 DQN 中所维护的经验回放池，保存了最近的轨迹数据。

然而式 (2.25) 中的 $\max_{a'} Q_{\phi_{\text{targ}}}(s',a';)$ 中的 max 操作在连续动作空间下难以进行，因此 DDPG 还学习了一个确定性策略 μ_θ，该策略直接输出最优动作 $a^* = \mu_{\theta_{\text{targ}}}(s)$，因此

$$\max_{a'} Q_{\phi_{targ}}(s',a') \approx Q_{\phi_{\text{targ}}}\left(s',\mu_{\theta_{\text{targ}}}(s')\right). \tag{2.26}$$

上述为动作值函数的更新过程，策略则通过最大化该策略对应动作值函数在状态 s 下的期望来更新：

$$\max_{\theta} \underset{s\sim D}{\mathrm{E}}\left[Q\left(s,\mu_\theta(s);\phi\right)\right]. \tag{2.27}$$

以上两个步骤迭代交互进行，则可学习确定性最优策略 μ_θ。算法6给出了 DDPG 训练的一般流程。

后一章节所述的 MADDPG[47] 是以 DDPG 为基础的一种多智能体版本。

Algorithm 6: DDPG 算法流程

Input: 初始策略参数 θ，Q 函数参数 ϕ，空的经验缓冲池 $\mathcal{D}$

设置目标网络参数为主网络参数 $\theta_{\text{targ}} \leftarrow \theta, \phi_{\text{targ}} \leftarrow \phi$;

while 策略尚未收敛 **do**

观测状态 s，根据 $a = clip(\mu_\theta(s) + \epsilon, a_{low}, a_{high})$，其中 $\epsilon \sim \mathcal{N}$;

在环境中执行动作 a;

观测下一状态 s'，奖励 r，以及轨迹终止信号 d;

将 (s, a, r, s', d) 存入 $\mathcal{D}$ 中;

if s' 是终止状态 **then**

则重置环境状态;

for 更新次数 **do**

从 $\mathcal{D}$ 中随机采样批量 $B = \{(s, a, r, s', d)\}$;

计算目标:

$$y(r, s', d) = r + \gamma(1 - d)Q_{\phi_{\text{targ}}}\left(s', \mu_{\theta_{\text{targ}}}(s')\right)$$

通过梯度下降更新 Q 函数:

$$\nabla_\phi \frac{1}{|B|} \sum_{(s,a,r,s',d) \in B} \left(Q_\phi(s, a) - y(r, s', d)\right)^2$$

使用梯度上升更新策略:

$$\nabla_\theta \frac{1}{|B|} \sum_{s \in B} Q_\phi\left(s, \mu_\theta(s)\right)$$

更新目标网络:

$$\phi_{\text{targ}} \leftarrow \rho\phi_{\text{targ}} + (1 - \rho)\phi$$

$$\theta_{\text{targ}} \leftarrow \rho\theta_{\text{targ}} + (1 - \rho)\theta$$

第三章　多智能体强化学习协作算法

本节中首先介绍多智能体强化学习建模的一般框架——Dec-POMDP[46]，然后介绍多智能体强化学习协作中的一些算法，这些算法包括：1) 基于独立学习的独立 Actor-Critic (Independent Actor-Critic, IAC) 和独立 Q 学习 (Independent Q-learning, IQL)[45]，2) 解决多智能体奖励分配问题的 COMA 算法[50]，3) 混合竞争合作场景下的 MADDPG[47] 算法，以及 4) 基于值分解方法的值分解网络 (Value Decomposition Networks, VDN)[48]、QMIX[41] 以及 QTRAN[49]。

3.1　Dec-POMDP

Dec-POMDP 是一个用以建模多智能体强化学习协作任务框架[46]，是基于 MDP[1] 以及 POMDP[46] 的一种多智能体情形下的拓展形式。本章后续所介绍的多智能体强化学习协作算法均基于 Dec-POMDP 进行建模。因此这里做简要概述。

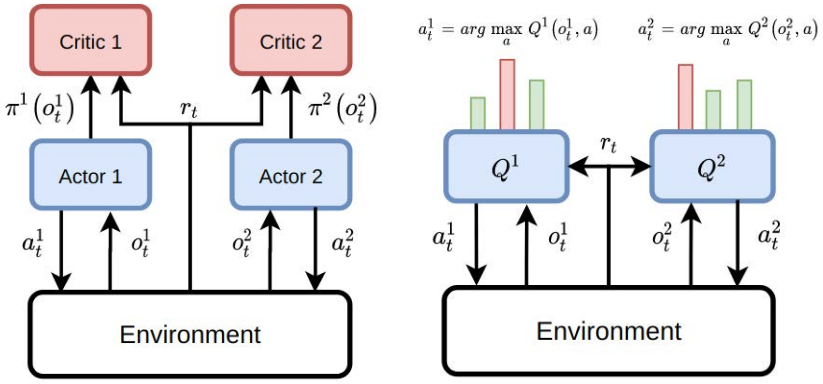
具体来说，Dec-POMDP 定义为元组 $\mathcal{G} = \langle \mathcal{S}, \mathcal{A}, \mathcal{P}, r, \mathcal{Z}, \mathcal{O}, N, \gamma \rangle$，其中 $s \in \mathcal{S}$ 表示环境的真实状态，对于智能体来说一般不可知。$\mathcal{A}$ 是智能体的动作空间，$\gamma \in [0, 1]$ 是折现因子。在每一时刻，所有智能体 $i \in \{1, 2, \cdots, N\}$ 均选取一个动作 $a^i \in \mathcal{A}$，构成一个联合动作 $\mathbf{a} = \left\{a^1, a^2, \cdots, a^N\right\} \in \mathcal{A}^N$ 向量。接下来环境通过状态转移函数 $\mathcal{P}(s'|s, \mathbf{a}) : \mathcal{S} \times \mathcal{A}^N \times \mathcal{S} \mapsto [0, 1]$ 进入下一个状态 s'。所有智能体共享奖励函数 $r(s, \mathbf{a}) : \mathcal{S} \times \mathcal{A}^N \mapsto \mathbb{R}$。部分可观测的情况下，其中每个智能体从观测函数 $\mathcal{Z}(s, i) : \mathcal{S} \times N \mapsto \mathcal{O}$ 获得部分观测 $\mathcal{Z}(s, i) : \mathcal{S} \times N \mapsto \mathcal{O}$。每个智能体 i 还具有观测行动历史 $\tau^i \in \mathcal{T} \equiv (\mathcal{O} \times \mathcal{A})^*$。

3.2　独立学习

独立学习 (Independent Learning, IL) 代表一类多智能体强化学习算法，表示该类算法使用传统单智能体强化学习的模式为每一个智能体单独学习一个策略，所有策略在训练阶段彼此独立，不进行任何信息的共享。具体地，根据独立学习所使用的算法而言，又可分为独立 Actor-Critic 学习 (Independent Actor Critic, IAC)[50] 和独立 Q 学习 (Independent Q-learning, IQL)[45] 等。

IAC[50] 是 Actor-Critic[63] 算法的一种多智能体版本。考虑为每一个智能体单独学习一个策略 actor 以及用于评估 actor 性能的 critic，actor 和 critic 均只依赖于当前智能体自身的观测和动作，不使用其他智能体的任何信息。

IQL[45] 是 Q-learning[64] 的一种多智能体版本。IQL 中的每一个智能体单独学习一个 Q^i 函数，同样该 Q^i 函数仅依赖于当前智能体自身的观测和动作。将 DQN 中的深度网络以及经验回放等机制也可以嵌入到 IQL 中，使得 IQL 变为一种深度多智能体强化学习算法[69]，在两人乒



(a) IAC 结构示意图。　　(b) IQL 结构示意图。

图 3.1　IAC 和 IQL 结构示意图。在训练阶段智能体之间的学习彼此独立，不进行任何信息的共享，与单智能体强化学习相比，仅仅是智能体数量的增加。

乒球游戏中[69]，这种方法取得了较好的经验结果。

IAC 和 IQL 这种学习时的相互独立性保证了算法具有较好的可拓展性，即这种完全分散的设定可以应用于任意智能体个数的场景下。然而，他们的缺点也显而易见，他们未能处理由于多个智能体策略的变化所导致的环境的非静态性问题[44]，因此该方法在有限探索的条件下并不具有收敛性保证[39]。另外，独立学习的模式也没有在合作场景下进行奖励的分配[50]。在实践中，IAC 与 IQL 一般作为评估多智能体强化学习算法的基准算法[69,70]。图3.4中给出了 IAC 和 IQL 结构示意图。

3.3　COMA

COMA[50] 是一种基于 Actor-Critic[51] 结构的多智能体强化学习算法，主要解决多智能体奖励分配[52] 问题。在合作场景中，环境对于联合动作通常仅反馈以联合奖励，这使得每个智能体很难推断自己对联合奖励的贡献。COMA 在这种设定下，基于差分奖励[53] 的思想，使用反事实基准 (counterfactual baseline) 来计算每个策略的优势函数，该优势函数最终用以计算每个智能体的策略梯度。

在多智能体情况下，最简单的计算策略梯度的方式为：

$$g = \nabla_{\theta^\pi} \log \pi\left(i|\tau_t^i\right)\left(r + \gamma V\left(s_{t+1}\right) - V\left(s_t\right)\right). \tag{3.1}$$

然而这种方法无法解决奖励分配的问题，由于计算 TD 误差时只考虑了全局奖励，因此为每一个 actor 计算的梯度未能表现出该智能体对于整体的贡献。因此 COMA 使用了一种反事实的基准，使用如下差分奖励的思路计算每一个智能体对于整体的贡献：

$$D^i = r(s, \mathbf{a}) - r\left(s, \left(\mathbf{a}^{-i}, c^i\right)\right), \tag{3.2}$$

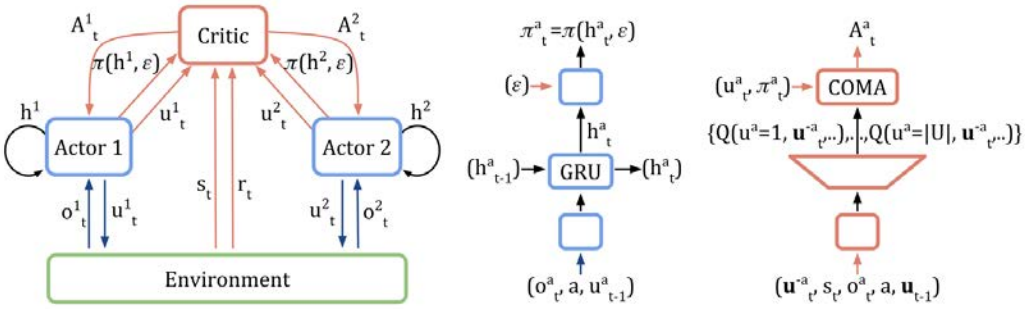


图 3.2　COMA 网络结构示意图。左侧为智能体与环境整体交互示意图；中间为智能体策略网络结构示意图，每个智能体策略网络基于 RNN；右侧为集中的 critic 网络结构。

其中 D^i 为智能体 i 的差分奖励，c^i 为智能体 i 的默认动作。该式直观地表达出智能体 i 的奖励为整体奖励减去掉智能体 i 采取默认动作获得的奖励。使用类似的想法，可以计算每一个智能体的优势函数：

$$A^i(s,\mathbf{a}) = Q(s,\mathbf{a}) - \sum_{a'^i} \pi^i\left(a'^i \mid \tau^i\right) Q\left(s,\left(\mathbf{a}^{-i},a'^i\right)\right), \tag{3.3}$$

其中 $b(s,\mathbf{a}^{-i}) = \sum_{a'^i} \pi^i\left(a'^i \mid \tau^i\right) Q\left(s,\left(\mathbf{a}^{-i},a'^i\right)\right)$ 为反事实基准，$\mathbf{a}^{-i}$ 除去智能体 i 后其他智能体的联合动作。因此 COMA 的策略梯度可以写成：

$$g_k = \mathbb{E}_{\boldsymbol{\pi}}\left[\sum_i \nabla_{\theta_k} \log \pi^i\left(a^i \mid \tau^i\right) A^i(s,\mathbf{a})\right]. \tag{3.4}$$

有了这样的梯度信息，即可使用梯度上升更新策略。

3.4　MADDPG

MADDPG[47] 是一种基于 DDPG[54] 的一种多智能体强化学习算法，其主要想法是在训练阶段学习一个集中价值函数，通过增加其他智能体的额外信息作为其输入，以增强集中价值函数的学习，同时缓解由于其他智能体策略变化产生的环境不稳定问题。而在执行阶段，学习的策略仅使用每个智能体的局部观测信息，以保证策略可以分散执行。这也是 CTDE 范式的主要思想，图3.3给出了 MADDPG 的结构示意图。

下面简述 MADDPG 的策略学习方法。假设智能体 i 的确定性策略为 μ^i，参数为 θ_i，经验回放池为 $\mathcal{D}$，类比 DDPG，智能体 i 策略的梯度为：

$$\nabla_{\theta_i} J\left(\theta_i\right) = \mathbb{E}_{s,\mathbf{a}\sim\mathcal{D}}\left[\nabla_{\theta_i}\mu^i\left(o^i\right)\nabla_{a^i} Q_{\phi_i}\left(s,\mathbf{a}\right)\big|_{a^i=\mu^i(o^i)}\right], \tag{3.5}$$

其中经验回放池 $\mathcal{D} = \{(s,\mathbf{a},s',r^1,r^2,\cdots,r^N)\}$ 存储了智能体 μ^i 过去与环境交互一段时间的轨迹数据，$Q_{\phi_i}(s,\mathbf{a})$ 则是智能体 i 的联合动作值函数，其输入是环境的全局状态以及所有智能体的联合动作 $\mathbf{a}$。

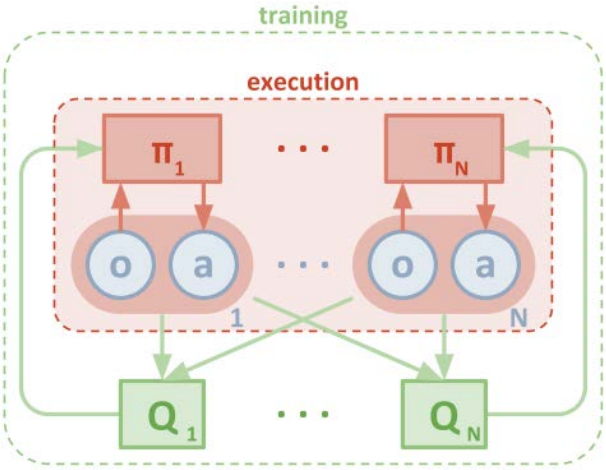


图 3.3　MADDPG 网络结构示意图。红色部分的模块为智能体策略，在执行阶段分散执行。绿色模块为联合动作值函数，仅在训练阶段使用，其输入为所有智能体的联合观测与联合动作。

假设智能体 i 的联合动作值函数为 Q_{ϕ_i}，其更新方法如下：

$$\mathcal{L}(\phi_i) = \mathbb{E}_{s,\mathbf{a},r,s'}\left[\left(Q_{\phi_i}(s,\mathbf{a}) - y\right)^2\right], \tag{3.6}$$

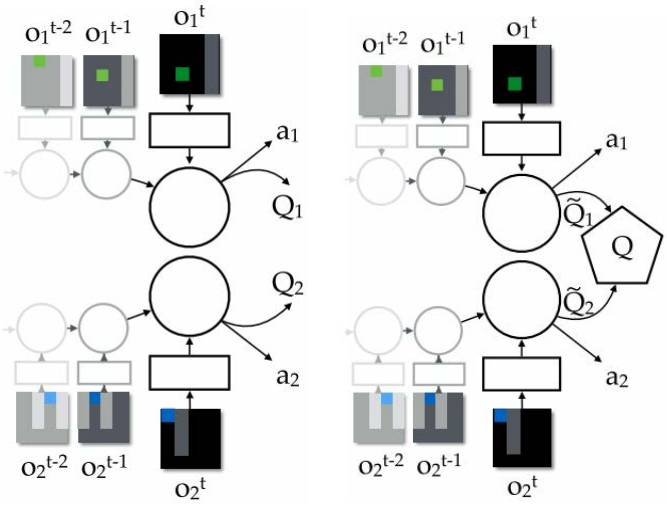
其中 $y = r^i + \gamma Q_{\text{targ}_i}(s',\mathbf{a}')\big|_{a^{j'}=\boldsymbol{\mu}^{j'}(o^{j'})}$，$Q_{\text{targ}_i}$ 是智能体 i 的目标网络。

可以看到，MADDPG 的更新方式与 DDPG 无太大差异，主要的不同在于联合动作值函数的输入为所有智能体的联合动作，且在更新智能体策略参数及其联合动作值函数时，需固定其他智能体的动作为从经验回放池中采样的动作。

另外，可以发现 MADDPG 中联合动作值函数的学习受到智能体数量的约束，也就是说，这种学习联合动作值函数的方式需要将所有智能体的观测值和动作作为输入，输入规模随智能体数量的增加呈线性增长，当智能体数量增加到一定程度时，值函数的学习难度将会非常大。

3.5　值分解网络

于 IQL 不同，值分解网络 (Value Decomposition Networks, VDN)[48] 旨在学习一个联合 Q_{tot} 函数，使用该联合 Q_{tot} 函数对所有智能体的联合动作进行状态动作值函数评估。直觉上，在 N



(a) IQL 结构示意图。　　(b) VDN 结构示意图。

图 3.4　IQL 和 VDN 结构对比示意图。

个智能体的环境中，联合动作价值函数 $Q_{tot}(s, \mathbf{a})$ 可以表达为：

$$Q_{tot}(s, \mathbf{a}) = \mathbb{E}[\sum_{t=0}^{T} \gamma^t r_{tot}] \tag{3.7}$$

$$= \mathbb{E}\Big[\sum_{t=0}^{T} \gamma^t (r^1 + r^2 + \cdots + r^n)\Big] \tag{3.8}$$

$$= \mathbb{E}\Big[\sum_{t=0}^{T} \gamma^t r^1\Big] + \mathbb{E}\Big[\sum_{t=0}^{T} \gamma^t r^2\Big] + \cdots + \mathbb{E}\Big[\sum_{t=0}^{T} \gamma^t r^N\Big] \tag{3.9}$$

$$= Q^1 + Q^2 + \cdots + Q^N, \tag{3.10}$$

其中 Q_{tot} 是集中 Q 值函数，Q^i 是每一个智能体的 Q 函数，r^i 是智能体 i 实际的奖励值，r_{tot} 是所有智能体执行联合动作 $\mathbf{a}$ 的联合奖励值。

通过假设联合奖励可以线性地表达为所有智能体各自奖励值的和，可得到联合动作 Q_{tot} 与 Q^i 存在下面的关系：

$$Q_{tot}(\boldsymbol{\tau}, \mathbf{a}; \theta) = \sum_{i=1}^{N} Q^i(\tau^i, a^i; \theta^i), \tag{3.11}$$

其中 τ^i 为智能体 i 的动作观测序列。图3.4(b) 给出了 VDN 结构示意图。

类似 DQN 的目标函数，通过端到端地最小化如下联合动作 Q 函数的损失函数，即可学得每个智能体的策略。

$$\mathcal{L}(\theta) = \sum_{i=1}^{b} \left(r_{tot} + \gamma \max_{\mathbf{a}} Q_{tot}(\boldsymbol{\tau}, \mathbf{a}; \theta^-) - Q_{tot}(\boldsymbol{\tau}, \mathbf{a}; \theta)\right)^2. \tag{3.12}$$

其中 $Q_{tot}(\boldsymbol{\tau}, \mathbf{a}; \theta^-)$ 为目标网络。

VDN 的主要问题在于: 1) 过于简化联合动作值函数的表示形式。由于 r_{tot} 并不一定可以表示为所有所有智能体各自奖励值之和，因此 VDN 所依赖的这种假设能否成立取决于问题的具体形式；2) VDN 在学习过程中没有很好地利用全局状态信息辅助训练。

3.6 基于值分解的单调混合网络

QMIX[41] 通过强制 $\frac{\partial Q_{tot}}{\partial Q^i} \geq 0, i \in \{1, 2, \cdots, N\}$，将 VDN 的加性值分解扩展到式 (3.13) 中更一般的情况，VDN[48] 可以看做所有的组合系 $\alpha_i, i = 1, 2, \cdots, N$ 都被设置为 1 的一种特殊情况。

$$Q_{tot}(\boldsymbol{\tau}, \mathbf{a}; \theta) = \sum_{i=1}^{N} \alpha_i Q^i(\tau^i, a^i; \theta^i), \alpha_i \geq 0, \tag{3.13}$$

为了保证式 (3.13) 成立，QMIX 需要保证下式中的单调性条件成立:

$$\frac{\partial Q_{tot}}{\partial Q^i} \geq 0, \forall i \in \{1, 2, \cdots, N\}. \tag{3.14}$$

为了学习组合系数 $\alpha_i, i = 1, 2, \cdots, N$，QMIX 采用了一种基于超网络[?] 的混合网络结构，该结构以复杂的非线性方式确保集中和分散策略之间的一致性。同时，通过限制混合网络的权值为正值来保证式 (3.14) 中的约束成立。图3.5给出了 QMIX 结构示意图。

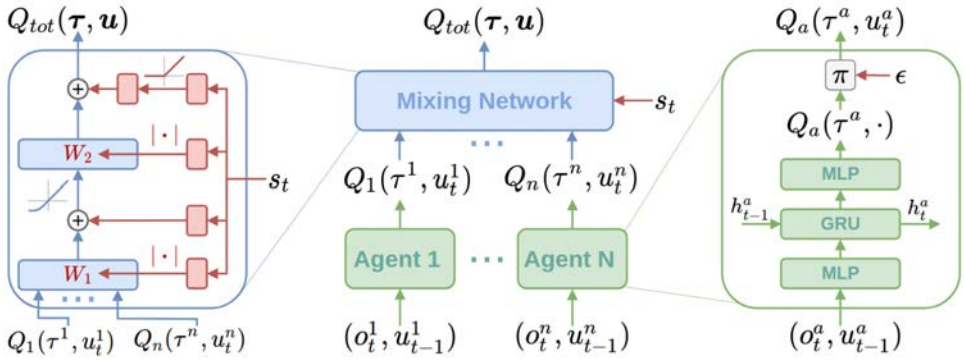


图 3.5 QMIX 网络结构示意图。左侧为基于超网络的混合网络结构；中间为 QMIX 整体结构，由 N 个智能体的策略网络以及混合网络结构构成；右侧为每一个智能体策略网络结构，智能体的策略使用基于 RNN 的递归网络。

3.7 QTRAN

之前讨论的 VDN 以及 QMIX 是值函数分解这一类多智能体强化学习的代表，VDN 将联合动作值函数分解为每个智能体单个动作值函数的之和，而 QMIX 则将 VDN 这种加性分解拓展到一种更加复杂的非线性单调分解方式，这种非线性方式使得 QMIX 学习的联合动作价值函数具有更大的假设空间。然而 QMIX 这种值分解技术依然受到结构上的约束，在某些任务中可能导致其无法学习到有用的策略。

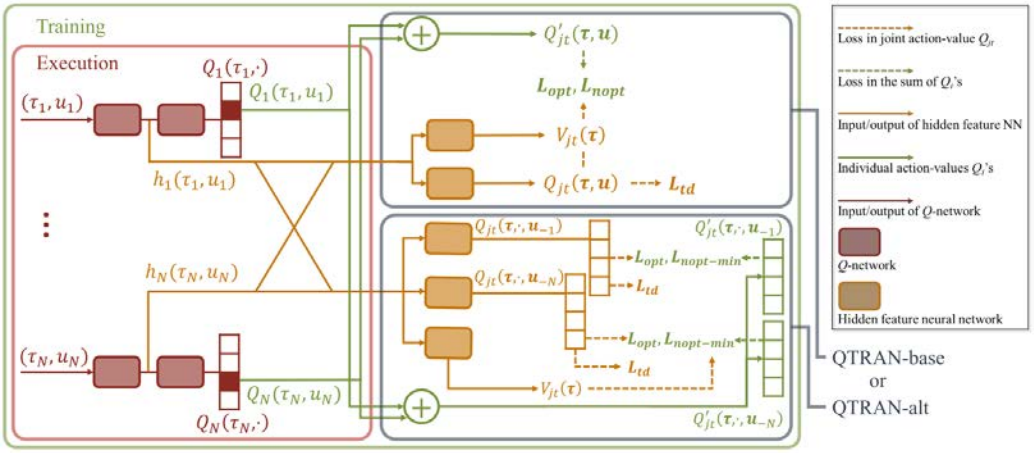


图 3.6　QTRAN 结构示意图。

基于 QMIX 所存在的结构约束，QTRAN[49] 提出学习一个集中式无约束的联合 Q 函数以及一个 VDN 分解的联合 Q 函数。因为无约束的 Q 值无法被有效最大化，VDN 分解的 Q 值被用来产生近似最大的联合动作。这使得 QTRAN 能够表示比 QMIX 假设空间更大的联合 Q 函数。

但在具体的实践中，QTRAN 由于受其优化方法的限制，在复杂环境中的性能并不理想。

第四章　基于多步异策略的多智能体协作学习

本章介绍本文提出的基于多步异策略的多智能体协作学习方法——SMIX(λ)，该方法旨在解决多智能体强化学习中的集中价值函数估计问题。在正式介绍 SMIX(λ) 之前，首先介绍相关的问题设定以及相关的背景知识。

4.1　简介

近年来，强化学习在游戏领域[36] 和复杂的连续控制任务[37] 中都取得了巨大的成功。但是，许多现实世界中的问题中本质上是存在多个智能体的，例如网络数据包路由[71]，交通信号灯动控制[72]，社会困境以及多人博弈[73]，这些问题中，智能体数量的增多带来了单智能体情形中从未遇到的许多挑战。

特别地，在多智能体环境中的主要挑战包括以下几点：1）联合动作空间的大小随智能体的数量成指数增长[40,41]，2）由环境中多个智能体策略的改变导致的环境不稳定性[47,74] 以及 3）合作场景中的多智能体全局奖励分配问题[40,41]。这些挑战使得将所有智能体视为单个元智能体的完全集中方法，以及通过将其他智能体视为环境的一部分来分别训练每个智能体的完全分散式方法都变得非常困难[39–41,45]。

最近，在多智能体强化学习领域，由于概念的简单性和实用性，“集中训练、分散执行” (Centralized Training, Decentralized Execution, CTDE) 的范式变得非常流行[40,41,75,76]。它的关键思想是在训练期间学习所有智能体共享的集中价值函数，而每个智能体在执行阶段均以分散的方式部署执行。通过将集中价值函数充当每个智能体的环境，并结合适当的奖励分配机制，每个智能体各自的值函数可以被很方便地学习。

不幸的是，目前集中价值函数在 CTDE 方法中发挥的核心作用在当前领域中似乎并没有得到足够的重视[40,41,47,48]。多数工作通常用单智能体设定下的方式来学习集中价值函数，这导致在多智能体环境中会引入估计误差。另一方面，由于以下原因，估计多智能体环境中的集中价值函数本质上是困难的：1) 联合行动空间的“维度诅咒”问题导致的经验稀疏[55]；2) 环境非马尔可夫性质的挑战以及多智能体环境中的部分可观测性比在单智能体环境中更为严峻；3) 由于智能体之间的交互复杂，多智能体环境的复杂且难以建模。实际上，这些因素通常会导致集中价值函数不稳定，且估计的偏差和方差很高。

为了解决这些困难，本文提出了一个新的具有较高样本效率，基于 CTDE 框架的多智能体强化学习方法，称为 SMIX(λ)。SMIX(λ) 通过基于异策略的集中价值函数学习方法改进了集中价值函数估计，该方法消除了在训练过程中显式依赖集中贪婪行为 (Centralized Greedy Behaviour,

CGB) 假设的需要，并且引入 λ-回报[1] 可以更好地平衡偏差和方差。SMIX(λ) 使用异策略学习机制是由重要性采样 (Importance Sampling) 驱动的，但是通过经验回放 (Experience Replay) 机制来实现，这种机制与先前的单智能体学习中的 Q(λ) 方法有着密切的联系。通过结合这些要素，SMIX(λ) 方法有效地提高了样本效率，并稳定了训练过程。以星际争霸多智能体挑战赛 (StarCraft Multi-Agent Challenge, SMAC)[58] 为基准，SMIX(λ) 可以在几乎所有的场景中都获得了最好的性能。此外，通过将现有的 CTDE 型多智能体强化学习算法的集中价值函数估计器替换为本文新提出的 SMIX(λ)，观察到现有 CTDE 型多智能体强化学习算法可以获得显著的性能提升。

4.2 背景知识

4.2.1 Dec-POMDP

本文考虑协作式多智能体任务，其一般可以描述为 Dec-POMDP[46]。该任务一般定义为元组 $\mathcal{G} = \langle \mathcal{S}, \mathcal{A}, \mathcal{P}, r, \mathcal{Z}, \mathcal{O}, N, \gamma \rangle$，其中 $s \in \mathcal{S}$ 表示环境的真实状态，对于智能体策略来说一般不可知。$\mathcal{A}$ 是单个智能体的动作空间，$\gamma \in [0, 1]$ 是折现因子，表现了未来奖励随时间的折损速率。在每个时间步，所有智能体均选择一个动作 $a^i \in \mathcal{A}$，所有智能体选择的动作构成了一个联合动作向量 $\mathbf{a} = \left\{a^1, a^2, \cdots, a^N\right\} \in \mathcal{A}^N$ 。接下来环境通过状态转移函数 $\mathcal{P}(s'|s, \mathbf{a}) : \mathcal{S} \times \mathcal{A}^N \times \mathcal{S} \mapsto [0, 1]$ 转移到下一个状态 s'。所有智能体具有相同的奖励函数 $r(s, \mathbf{a}) : \mathcal{S} \times \mathcal{A}^N \mapsto \mathbb{R}$，所得奖励是对所有智能体联合动作的一个反馈，而非单个智能体的奖励。本文考虑部分可观测的情形，其中每个智能体从观测函数 $\mathcal{Z}(s, i) : \mathcal{S} \times N \mapsto \mathcal{O}$ 获得部分观测 $\mathcal{Z}(s, i) : \mathcal{S} \times N \mapsto \mathcal{O}$。每个智能体 i 还具有观测行动历史 $\tau^i \in \mathcal{T} \equiv (\mathcal{O} \times \mathcal{A})^*$。随机策略是一个映射，其定义为：$\pi(a|\tau) : \mathcal{T} \times \mathcal{A} \mapsto [0, 1]$。

在 CTDE 范式的训练阶段，一个集中的值函数 $Q([s, \boldsymbol{\tau}], \mathbf{a})$(或简单表示为 $Q(\boldsymbol{\tau}, \mathbf{a})$) 是从所有智能体的局部观测历史以及全局状态学到的。在执行阶段，每个智能体的策略 π^i 只依赖于其局部观测历史 τ^i。

4.2.2 集中价值函数的假设空间

假设空间 $\mathcal{H}$ (或假设集) 是一个用于将输入空间映射到输出空间的所有可能假设所构成的空间[77,78]。特别是在多智能体系统中，所有智能体的联合行动空间随着智能体数量的增加而呈指数增加，这意味着 CVF 的假设空间应该足够大以处理这种复杂性。此外，为了使得每个智能体在不依赖集中价值函数的情况下就可以基于其局部观测进行决策，通常假设以下集中贪婪行

为 (Centralized Greedy Behavior，CGB) 成立：

$$\underset{\mathbf{a}}{\operatorname{argmax}}\, Q_{tot}(\boldsymbol{\tau}, \mathbf{a}) = \begin{pmatrix} \underset{a^1}{\operatorname{argmax}}\, Q^1\left(\tau^1, a^1\right) \\ \vdots \\ \underset{a^N}{\operatorname{argmax}}\, Q^N\left(\tau^N, a^N\right) \end{pmatrix}. \tag{4.1}$$

这种性质建立了集中价值函数和分散式价值函数之间的结构约束，可以将其视为执行阶段简化了的奖励分配机制。

4.2.3 VDN, QMIX 以及 QTRAN

等式 (4.1) 的一个充分条件是下面的非负线性组合：

$$Q_{tot}(\boldsymbol{\tau}, \mathbf{a}; \theta) = \sum_{i=1}^{N} \alpha_i Q^i(\tau^i, a^i; \theta^i), \alpha_i \geq 0, \tag{4.2}$$

其中 Q_{tot} 是集中 Q 值函数，Q^i 是每一个智能体的 Q 函数。在 VDN[48] 中，所有的组合系 $\alpha_i, i = 1, 2, \cdots, N$ 被设置为 1。QMIX[41] 通过强制 $\frac{\partial Q_{tot}}{\partial Q^i} \geq 0, i \in \{1, \cdots, N\}$ 将这个加性值分解扩展到更一般的情况。下面的定理明确给出了 QMIX 施加在集中价值函数假设空间上的相应的结构约束。

定理 1. *对于 QMIX, 如果 $\frac{\partial Q_{tot}}{\partial Q^i} \geq 0\ \forall i \in \{1, 2, \cdots, N\}$, 那么有*

$$\begin{aligned} &\max_{\mathbf{a}} Q_{tot}(\boldsymbol{\tau}, \mathbf{a}) = \\ &Q_{tot}(\boldsymbol{\tau}, \underset{a^1}{\operatorname{argmax}}\, Q^1(\tau^1, a^1), \cdots, \underset{a^N}{\operatorname{argmax}}\, Q^N(\tau^N, a^N)). \end{aligned} \tag{4.3}$$

注意，VDN 和 QMIX 算法均依赖此结果来简化其优化过程。QTRAN 进一步放宽了 VDN 和 QMIX 的约束，其假设空间为式(4.1)的充分必要条件所构成的更大假设空间，但代价是要在所有智能体构成的联合动作空间中对联合价值函数进行优化，即使在智能体数量较少的情况下，这也是一个不小的挑战。虽然在 QTRAN 中提出了一种坐标下降型方法来解决这个问题，但是这种方法的可扩展性和实际使用范围可能会受到限制。

4.3 方法

在本节中，首先给出 SMIX(λ) 方法的细节。SMIX(λ) 是一个 SARSA(λ)[79] 风格的异策略方法，旨在 CTDE 框架内学习一个更好的集中价值函数。

4.3.1 在训练阶段放松 CGB 假设

在标准的 CTDE 方法中，首先训练所有智能体的集中函数，然后将其值分配给单个智能体，以指导每个智能体的训练过程。该思想的一个典型实现是 QMIX[41]，其中集中的函数是通过传

统的 Q-learning 算法来学习的。然而，由于联合动作空间的高维度，对集中价值函数取 max 操作是无法实现的。为了解决这个问题，一般遵循前面提到的 CGB 假设，尽管它看起来是不现实的。[1]

可以使用基于 SARSA[1] 的方法代替 Q-learning，其中使用 Bellman 期望操作符来学习 Q_{tot} 函数。如前面章节所述，SARSA 是一种 on-policy 方法，只考虑单步回报。接下来，本文将把这个方法扩展到异策略设定下，并将其与多步回报集成起来以处理集中价值函数估计中的方差和偏差问题。

4.3.2 无重要性采样的异策略学习

缓解联合动作空间的维度诅咒，并提高探索能力的途径之一是异策略学习。用 $\boldsymbol{\mu}$ 表示行为策略 (behavior policy)，$\boldsymbol{\pi}$ 表示目标策略 (target policy)，一般使用 $\boldsymbol{\mu}$ 生成的数据 $\boldsymbol{\tau}$ 来评估策略 $\boldsymbol{\pi}$ 的 Q 值函数是通过以下的方法：

$$Q(\boldsymbol{\tau}, \mathbf{a}) \leftarrow Q(\boldsymbol{\tau}, \mathbf{a}) + \mathbb{E}_{\boldsymbol{\mu}}\left[\sum_{t \geq 0} \gamma^t \left(\prod_{i=1}^{t} \rho_i\right) \delta_t^{\boldsymbol{\pi}}\right], \tag{4.4}$$

其中每一个 ρ_i 是一个非负系数，并且满足当 $t = 0$ 时，$\prod_{i=1}^{t} \rho_i = 1$。误差项 $\delta_t^{\boldsymbol{\pi}}$ 一般写成下面的期望差分误差[80]，

$$\delta_t^{\boldsymbol{\pi}} = r_{t+1} + \gamma \mathbb{E}_{\boldsymbol{\pi}} Q\left(\boldsymbol{\tau}_{t+1}, \cdot\right) - Q\left(\boldsymbol{\tau}_t, \mathbf{a}_t\right), \tag{4.5}$$

其中 $\mathbb{E}_{\boldsymbol{\pi}} Q(\boldsymbol{\tau}, \cdot) = \sum_{\mathbf{a}} \boldsymbol{\pi}(\mathbf{a}|\boldsymbol{\tau}) Q(\boldsymbol{\tau}, \mathbf{a})$ 。特别地，对于重要性采样 (IS) 方法，将式 (4.4) 中的每个 ρ_i 定义为它们的轨迹在目标策略 $\boldsymbol{\pi}$ 和行为策略 $\boldsymbol{\mu}$ 下发生的相对概率，也称为重要性采样比率，也就是说，$\rho_i = \frac{\boldsymbol{\pi}(\mathbf{a}_i|\boldsymbol{\tau}_i)}{\boldsymbol{\mu}(\mathbf{a}_i|\boldsymbol{\tau}_i)}$。

尽管重要性采样方法具有理论保证，但其在多智能体环境中面临着巨大的挑战：1) 上式中的联合乘积会造成方差变大[81], 2) 当智能体数量增加时，联合动作空间中的“维度诅咒”问题使之在仅有单个时间步 i 的情况下仍然难以用于计算 $\boldsymbol{\pi}(\mathbf{a}_i|\boldsymbol{\tau}_i)$。文献[39] 中曾提出一种方法以解决第一个问题中轨迹上的乘积，但如何解决第二个问题依然是一个开放性问题。

以上分析强调了探索新的替代方案的必要性，即可以实现异策略学习而无需在多智能体设置中进行重要性采样的替代方法。

4.3.3 SMIX(λ) 方法

为了实现上述目标，SMIX(λ) 的关键思想是进一步简化式 (4.4) 中的系数 ρ_i，以减少重要性采样估计量的方差，并有可能绕过计算 $\boldsymbol{\pi}(\cdot|\boldsymbol{\tau})$ 的维数诅咒问题。

[1]注意，这个贪婪的假设不仅在 Q 学习算法的更新规则需要被使用，在经典的 n-step Q 学习方法和 Watkins's $Q(\lambda)$ 方法[79] 也都需要被使用。因此，为了在学习阶段消除对这种假设的依赖，在学习集中价值函数网络时，完全有必要放弃这种更新方法。

具体来说，根据式 (4.4) 放宽每个系数 $\rho_i = 1.0$，并使用经验回放机制来存储最新的异策略数据。事实上，过去的工作[82] 表明与环境交互过程中产生的异策略数据与当前策略的数据的分布高度相关。

接下来，使用 λ-回报[1] 作为 TD 目标估计量，其定义如下：

$$G_t^{\lambda} = (1-\lambda)\sum_{n=1}^{\infty}\lambda^{n-1}G_t^{(n)}, \tag{4.6}$$

其中 $G_t^{(n)} = r_{t+1} + \gamma r_{t+2} + \cdots + \gamma^n \mathbb{E}_{\boldsymbol{\pi}} Q(\boldsymbol{\tau}_{t+n}, \mathbf{a}_{t+n}; \theta^-)$ 为 n 步回报，θ^- 为目标网络的参数。将其代入式 (4.4)，并对于任意 i，设 $\rho_i = 1.0$, 有，

$$Q(\boldsymbol{\tau}, \mathbf{a}) \leftarrow Q(\boldsymbol{\tau}, \mathbf{a}) + \mathbb{E}_{\boldsymbol{\mu}}\left[\sum_{t\geq 0}\gamma^t (G_t^{\lambda} - Q(\boldsymbol{\tau}_t, \mathbf{a}_t))\right]. \tag{4.7}$$

在实现上, SMIX(λ) 以端到端的方式进行训练，集中价值函数 $Q_{tot}^{\boldsymbol{\pi}}$ 具有下列形式:

$$\mathcal{L}_t(\theta) = \sum_{i=1}^{b}\left[(y_i^{tot} - Q_{tot}^{\boldsymbol{\pi}}(\boldsymbol{\tau}, \mathbf{a}; \theta))^2\right], \tag{4.8}$$

其中 $y_i^{tot} = G_t^{\lambda}$ 在式 (4.6) 中定义且通过经验回放机制进行估计。

注意，在训练阶段，通过放弃 Q-学习更新规则来消除 CVF 和分散策略之间的明确结构约束，这使得 SMIX(λ) 在比 QMIX[41] 更大的假设空间中进行优化，SMIX(λ) 中 $Q_{tot}^{\boldsymbol{\pi}}(\boldsymbol{\tau}, \mathbf{a}; \theta)$ 的深层网络体系结构与 QMIX 相同，只是训练目标不同。这表明 SMIX(λ) 要求混合网络中的所有权重都是非负的，这是 CGB 假设的一个充分条件。因此，SMIX(λ) 的假设空间小于 QTRAN。然而，SMIX(λ) 的样本复杂度要比 QTRAN 好得多，因为 QTRAN 必须估计所有智能体在高维联合作用空间上的期望值，即使在少数智能体的情况下，这在计算上也是有挑战性的。

算法7提供了 SMIX(λ) 的一般训练过程。值得注意的是，本文提出的集中价值函数训练方法是一种通用的方法，可以很容易地应用于其他 CTDE 方法，如 VDN[48]、COMA[40]，甚至完全分散的方法，如 IQL[45]。

4.4 分析

在这一节中，首先在 SMIX(λ) 和 $Q(\lambda)$[57] 方法之间建立连接，对所提出的 SMIX(λ) 算法进行收敛性分析，该方法最初是为单智能体设置下的异策略值函数评估而提出的。

记 $G^{\boldsymbol{\pi}}$ 为 λ-回报估计量 (见式 (4.6)), 对于目标策略 $\boldsymbol{\pi}$ 的动作值，异策略方法的目标是使用行为策略 $\boldsymbol{\mu}$ 中的数据更正 $G^{\boldsymbol{\pi}}$，以满足以下标准，

$$\mathbb{E}_{\boldsymbol{\pi}}\left[G^{\boldsymbol{\pi}}\right] = \mathbb{E}_{\boldsymbol{\mu}}\left[G^{\boldsymbol{\mu},\boldsymbol{\pi}}\right], \tag{4.9}$$

其中 $G^{\boldsymbol{\mu},\boldsymbol{\pi}}$ 为纠正后的异策略回报。

Algorithm 7: SMIX(λ) 训练流程

初始化行为网络参数为 θ，目标网络参数为 θ^-, 初始经验缓冲池 $\mathcal{D}$ 容量为 $N_{\mathcal{D}}$, 训练批大小为 N_b;

for 每一个训练轨迹 **do**

  for 每一条轨迹 **do**

    for $t = 1$ *to* $T-1$ **do**

      获得所有智能体的部分观测 $\mathbf{o}_t = \{o_t^1, \cdots, o_t^N\}$ 以及全局状态 s_t;

      对所有智能体，根据其值函数 Q^i 以及 ϵ-greedy 规则选择动作 a_t^i;

      在环境中执行联合动作 $\mathbf{a}_t = \{a^1, a^2, \cdots, a^N\}$;

      获得奖励 r_{t+1}, 每一个智能体的下一部分观测 o_{t+1}^i 以及下一个全局状态 s_{t+1};

    将轨迹存入 $\mathcal{D}$, 若 $|\mathcal{D}| \geq N_{\mathcal{D}}$ 则使用新轨迹替换掉最旧的轨迹;

  从 $\mathcal{D}$ 中均匀采样 N_b 条轨迹;

  使用参数 θ^-，根据式 (4.6) 计算每一时刻的 λ-回报目标 y_i^{tot};

  通过最小化 $\sum_{t=1}^{T-1}\sum_{i=1}^{N_b}\left[(y_i^{tot} - Q_{tot}^{\boldsymbol{\pi}}(\boldsymbol{\tau}, \mathbf{a}; \theta))^2\right]$ 更新参数 θ;

  每隔 C 轨迹更新目标网络参数:$\theta^- \leftarrow \theta$;

计算 $G^{\boldsymbol{\mu},\boldsymbol{\pi}}$ 最常使用的方法是重要性采样 (Importance Sampling, IS), 其中重要性采样比率 ρ_i 饰演异策略矫正的角色，在 SMIX(λ) 中，该比率被放松为：$\rho_i = \frac{\boldsymbol{\pi}(\mathbf{a}_i|\boldsymbol{\tau}_i)}{\boldsymbol{\mu}(\mathbf{a}_i|\boldsymbol{\tau}_i)} = 0$，对应于式 (4.4), SMIX($\lambda$) 的更新规则可以表示为：

$$\begin{aligned} Q^{\text{SMIX}}(\boldsymbol{\tau}_t, \mathbf{a}_t) &\leftarrow Q^{\text{SMIX}}(\boldsymbol{\tau}_t, \mathbf{a}_t) + \mathbb{E}_{\boldsymbol{\mu}}\left[\sum_{k=t}^{\infty}\left(\prod_{i=t+1}^{k}\lambda\gamma\right)\delta_k^{\boldsymbol{\pi}}\right], \\ \delta_k^{\boldsymbol{\pi}} &= \left(r_{k+0} + \gamma\mathbb{E}_{\boldsymbol{\mu}}Q^{\text{SMIX}}(\boldsymbol{\tau}_{k+1}, \cdot) - Q^{\text{SMIX}}(\boldsymbol{\tau}_k, \mathbf{a}_k)\right). \end{aligned} \tag{4.10}$$

相对于异策略矫正的乘法操作，也可以使用一种加性操作[57]。特别地, 当计算 $G^{\boldsymbol{\mu},\boldsymbol{\pi}}$ 时，从式 (4.9) 中推导出地一个加性项被加入到每一个奖励值当中。这种加性矫正的主要优势在于不涉及重要性采样比率的乘积以及联合策略 $\boldsymbol{\pi}(\mathbf{a}|\boldsymbol{\tau})$, 因此完全避开了重要性采样方法的局限性[1]。具体地，$Q(\lambda)$ 方法的更新规则是[57]：

$$\begin{aligned} Q^{Q(\lambda)}(\boldsymbol{\tau}_t, \mathbf{a}_t) &\leftarrow Q^{Q(\lambda)}(\boldsymbol{\tau}_t, \mathbf{a}_t) + \mathbb{E}_{\boldsymbol{\mu}}\left[\sum_{k=t}^{\infty}\left(\prod_{i=t+1}^{k}\lambda\gamma\right)\hat{\delta}_k^{\boldsymbol{\pi}}\right], \\ \hat{\delta}_k^{\boldsymbol{\pi}} &= \left(r_{k+1} + \gamma\mathbb{E}_{\boldsymbol{\pi}}Q^{Q(\lambda)}(\boldsymbol{\tau}_{k+1}, \cdot) - Q^{Q(\lambda)}(\boldsymbol{\tau}_k, \mathbf{a}_k)\right). \end{aligned} \tag{4.11}$$

通过比较式 (4.10) 和式 (4.11), 可以发现 SMIX(λ) 与异策略 $Q(\lambda)$ 本质上是等价的，除了 SMIX(λ) 在 $\delta_k^{\boldsymbol{\pi}}$ 中计算 $\mathbb{E}_{\boldsymbol{\mu}}Q^{\text{SMIX}}(\boldsymbol{\tau}_{k+1}, \cdot)$，而 $Q(\lambda)$ 在 $\hat{\delta}_k^{\boldsymbol{\pi}}$ 中计算 $\mathbb{E}_{\boldsymbol{\pi}}Q^{Q(\lambda)}(\boldsymbol{\tau}_{k+1}, \cdot)$。

[1]但是当行为策略 $\boldsymbol{\mu}$ 应当与目标策略 $\boldsymbol{\pi}$ 足够接近的情况下，本文使用的经验回放机制将不会是一个问题 (见[82])。

注意 SMIX(λ) 是 $Q^{\pi}(\boldsymbol{\tau}, \mathbf{a})$ 的有偏估计, 然而 $Q(\lambda)$ 是无偏的。下面的定理表明当 $\boldsymbol{\pi}$ 和 $\boldsymbol{\mu}$ 充分接近时，SMIX(λ) 与 $Q(\lambda)$ 输出的差距是有界的。这表明 SMIX(λ) 与 $Q(\lambda)$ 算法一致。

定理 2. *假定根据 $Q_n^{SMIX}(\boldsymbol{\tau}_t, \mathbf{a}_t) = Q_n^{Q(\lambda)}(\boldsymbol{\tau}_t, \mathbf{a}_t)$ 更新价值函数, 其中 n 表示第 n 次更新。令 $\epsilon = \max_{\boldsymbol{\tau}} \|\boldsymbol{\pi}(\cdot|\boldsymbol{\tau}) - \boldsymbol{\mu}(\cdot|\boldsymbol{\tau})\|_1$, $M = \max_{\boldsymbol{\tau}, \mathbf{a}} |Q_n^{Q(\lambda)}(\boldsymbol{\tau}, \mathbf{a})|$。那么, $Q_{n+1}^{SMIX}(\boldsymbol{\tau}_t, \mathbf{a}_t)$ 和 $Q_{n+1}^{Q(\lambda)}(\boldsymbol{\tau}_t, \mathbf{a}_t)$ 之间的误差可以被下列表达式界限住:*

$$|Q_{n+1}^{SMIX}(\boldsymbol{\tau}_t, \mathbf{a}_t) - Q_{n+1}^{Q(\lambda)}(\boldsymbol{\tau}_t, \mathbf{a}_t)| \leq \frac{\epsilon\gamma}{1-\lambda\gamma} M. \tag{4.12}$$

证明: 首先，有

$$\begin{aligned} |\delta_t^{\boldsymbol{\pi}} - \hat{\delta}_t^{\boldsymbol{\pi}}| &= |\gamma \mathbb{E}_{\boldsymbol{\mu}} Q_n^{\text{SMIX}}(\boldsymbol{\tau}_{t+1}, \cdot) - \gamma \mathbb{E}_{\boldsymbol{\pi}} Q_n^{Q(\lambda)}(\boldsymbol{\tau}_{t+1}, \cdot)| \\ &= \gamma |\sum_{\mathbf{a}} \boldsymbol{\mu}(\mathbf{a}|\boldsymbol{\tau}_{t+1}) Q_n^{\text{SMIX}}(\boldsymbol{\tau}_{t+1}, \cdot) - \sum_{\mathbf{a}} \boldsymbol{\pi}(\mathbf{a}|\boldsymbol{\tau}_{t+1}) Q_n^{Q(\lambda)}(\boldsymbol{\tau}_{t+1}, \cdot)| \leq \gamma\epsilon M. \end{aligned}$$

因此

$$\begin{aligned} |Q_{n+1}^{\text{SMIX}}(\boldsymbol{\tau}_t, \mathbf{a}_t) - Q_{n+1}^{Q(\lambda)}(\boldsymbol{\tau}_t, \mathbf{a}_t)| &= \left| \mathbb{E}_{\boldsymbol{\mu}} \left[\sum_{k=t}^{\infty} \left(\prod_{i=t+1}^{k} \lambda\gamma \right) \delta_k^{\boldsymbol{\pi}} \right] - \mathbb{E}_{\boldsymbol{\mu}} \left[\sum_{k=t}^{\infty} \left(\prod_{i=t+1}^{k} \lambda\gamma \right) \hat{\delta}_k^{\boldsymbol{\pi}} \right] \right| \\ &= \left| \mathbb{E}_{\boldsymbol{\mu}} \left[\sum_{k=t}^{\infty} \left(\prod_{i=t+1}^{k} \lambda\gamma \right) \left(\delta_k^{\boldsymbol{\pi}} - \hat{\delta}_k^{\boldsymbol{\pi}} \right) \right] \right| \leq \left| \mathbb{E}_{\boldsymbol{\mu}} \left[\sum_{k=t}^{\infty} \left(\prod_{i=t+1}^{k} \lambda\gamma \right) (\gamma\epsilon M) \right] \right| \\ &\leq \mathbb{E}_{\boldsymbol{\mu}} \left[\frac{1}{1-\lambda\gamma} (\gamma\epsilon M) \right] = \frac{\epsilon\gamma}{1-\lambda\gamma} M. \end{aligned}$$

□

该定理表明 SMIX(λ) 与 $Q(\lambda)$ 在某些温和的条件下具有类型的收敛性质。下面的定理表明了 $Q(\lambda)$ 的收敛性质[57]。

定理 3. *考虑使用固定策略 $\boldsymbol{\pi}$ 和 $\boldsymbol{\mu}$ 从式 (4.11) 中学得的 Q 函数序列[57]。令 $\epsilon = \max_{\boldsymbol{\tau}} \|\boldsymbol{\pi}(\cdot|\boldsymbol{\tau}) - \boldsymbol{\mu}(\cdot|\boldsymbol{\tau})\|_1$。若 $\lambda\epsilon < \frac{1-\gamma}{\gamma}$, 那么在与 TD($\lambda$) 收敛一样的条件下，几乎可以保证:*

$$\lim_{n\to\infty} Q_n^{Q(\lambda)}(\boldsymbol{\tau}, \mathbf{a}) = Q^{\boldsymbol{\pi}}(\boldsymbol{\tau}, \mathbf{a}).$$

通过定理2 与定理3可知: 如果 $\boldsymbol{\pi}$ 与 $\boldsymbol{\mu}$ 足够接近, 那么 SMIX(λ) 可收敛到当前策略的价值函数 $Q^{\boldsymbol{\pi}}(\boldsymbol{\tau}, \mathbf{a})$。这对于多智能体强化学习可能具有重要的意义，因为它避免了重要性采样的缺陷。

上面的分析表明 SMIX(λ) 和 $Q(\lambda)$ 从形式上和分析上具有相似性。然而，将他们应用到多智能体强化学习中, 其计算复杂度本质上是不同的。这是由于为了计算加性矫正项，$Q(\lambda)$ 不得不计算式 (4.11) 中目标策略 $\boldsymbol{\pi}$ 的期望, 然而这在多智能体设置下是不现实的，因为联合动作空间维度随智能体数量指数增长。相反，SMIX(λ) 方法依赖经验回访机制来计算式 (4.10) 中的期望, 其计算复杂度仅随训练样本数量线性增长，而与联合动作空间以及智能体数量无关。这样的一种可拓展性使得本文的方法相较于 $Q(\lambda)$ 更适用于多智能体强化学习。

4.5 实验

这一节，首先介绍环境设置以及 SMIX(λ) 算法的实现细节，然后给出实验结果，并对视频回放中算法的行为进行分析，最后给出本算法集中价值函数的通用性实验结果以及各个模块的消融分析。

4.5.1 环境设置

本文在星际争霸多智能体挑战 (SMAC)[58] 平台对本文中的算法进行评估，该平台基于流行的实时战略游戏 (Real-Time Strategy, RTS) 星际争霸，提供了非常多丰富且具有挑战的协作对战地图以供研究者使用。需要注意的是，该平台关注于协作对战场景中的微操作 (micromanagement) 问题，即每个学习智能体控制一个单独的对战单元进行战斗。与之相对的称为宏观操作 (macromanagement), 该模式下所涉及宏观决策和经济管理 (例如建立兵营，收集资源等) 是该平台不考虑的。

在每个对战开始时，该平台首先将两组对战单元放置在地图中，各组内单元的初始位置随机分布，两组单元分别由多智能体强化学习算法和内置启发式游戏 AI 控制。启发式游戏 AI 使用内置的固定脚本令敌方单元攻击对方。在实验中，内置游戏 AI 的难度可手动设置，在本文的试验场景中默认设置为非常困难。任何一方单元全部被消灭或规定的时间到达则标志一局对战的终结。

该平台对环境的具体建模如下：

- **状态空间与观测空间**: 在每一个时间步，智能体 i 的观测值为其可视范围内的智能体特征，该特征包含了 `relative_x` (相对自身的横坐标距离), `relative_y` (相对自身的纵坐标距离), `distance` (相对自身的距离), `unit_type` (个体类型), `health` (生命值) 以及 `shield` (护甲) 等信息。状态信息则包括所有智能体的观测信息中心以及所有智能体相对于地图中心的坐标。
- **动作空间**: 每个智能体的动作包含 `attack[enemy id]`(攻击某智能体), `move[direction]` (移动到不同的方向), `stop`, `noop`(仅血量为 0 的智能体可采取该动作)。
- **奖励函数**: 在杀死敌方单位后可获得正奖励值，而当同盟方单位被杀死后得到负奖励值, 此外，赢得战斗可以得到额外的奖励值。具体的奖励值可自定义设置，本文实验中的奖励值设置对所有算法保持一致。

有关环境的完整详细信息，请参考[58]。

在本文的实验中，考虑下面三种类型的对战地图：其中 3m 与 8m 地图仅包含同类型智能体, 且我方和敌方智能体个数对等，属于同类且对称的地图类型；2s3z 和 3s5z 地图包含个体类型不同的智能体，但敌我方个体数量一致，属于异种且对称的地图类型；3s_vs_3z 考验算法能

表 4.1　本文实验中所使用的地图场景。

地图	我方兵种	敌方兵种	地图类型
3m	3 Marines	3 Marines	同类且对称
8m	3 Marines	3 Marines	同类且对称
2s3z	2 Marines & 3 Zealots	2 Marines & 3 Zealots	异类且对称
3s5z	3 Marines & 5 Zealots	3 Marines & 5 Zealots	异类且对称
3s_vs_3z	3 Marines	3 Zealots	微技能：风筝 (kiting)
2s_vs_1sc	2 Stalkers	1 Spine Crawler	微技能：交替开火

否学会风筝 *(kiting)* 技能，即通过“打一下就跑”的策略使得只能近身攻击的兵种无法对自身造成伤害；2s_vs_1sc 则考验算法能否学习到交替开火的微技能，以吸引敌方的注意力。总的来说，通过这些不同的对战地图，可以衡量算法学得复杂协作技能的能力。

本文中使用测试胜率 *(test win rate)* 作为算法性能的评估指标[58]。该评估指标通过以下评估流程计算：(1) 每 20000 个时间步暂停训练，然后每个智能体通过贪心地选择可以令其 Q 函数最大的动作, 以此独立地执行，直到收集到 24 条轨迹; (2) 统计在这 24 条轨迹中我方算法控制的智能体在规定的时间限制中打败敌方的次数 m；(3) 将 $m/24$ 作为最终胜率。

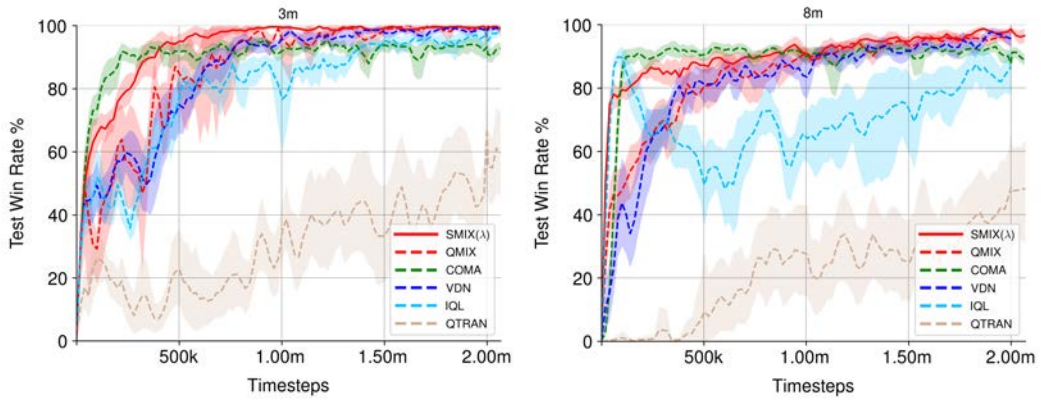
4.5.2　算法实现细节

SMIX(λ) 使用与 QMIX[41] 类似的架构。所有智能体的策略网络是一个共享的 DRQN (Deep Recurrent Q-Network)[83] 网络，其中递归神经网络层使用 64 维的 GRU[84]，此之前和此后均有一个全连接层。通过将每个智能体的 one-hot 向量加入到观测值中，以实现不同智能体使用共享网络但策略不同。探索策略使用 ϵ-greedy 对所有动作的 Q 值进行选择，训练阶段，ϵ 在 50k 个时间步中线性地从 1.0 退火为 0.05，此后采取固定的 $\epsilon = 0.05$ 进行动作选择。SMIX(λ) 的混合网络的架构与 QMIX[41] 相同。

与 QMIX[41] 不同的是，SMIX(λ) 使用 λ-回报来对集中价值函数进行估计 ($\lambda = 0.8$)。在轨迹采样阶段，本文并行地采样 4 条轨迹，并将其放入大小为 1500 的经验回放池中，经验回放池中的数据以轨迹的形式按条存放。训练时，每次均匀地从该经验回放池中采样 32 条轨迹用以训练。每收集 200 条轨迹，将当前策略 Q 网络的参数复制到目标网络，以更新目标网络。详细网络及训练参数见表4.2。

表 4.2　本文实验中默认使用的超参数。

参数	值	备注
γ	0.99	奖励折扣因子
batch size	32	训练批大小
buffer size	1500	经验回放池大小
学习率	0.0005	学习率
隐藏层维度	64	
网络层数	3	
混合网络层数隐藏层维度	32	
目标网络更新频率	200	每 200 条轨迹更新一次
优化器	RMSProp	
λ	0.8	λ-回报参数
初始 ϵ	1.0	初始 ϵ-greedy 的探索率
ϵ 退火率	50k	ϵ-greedy 退火速率
结束 ϵ	0.05	最终 ϵ-greedy 的探索率



(a) 3m　　(b) 8m

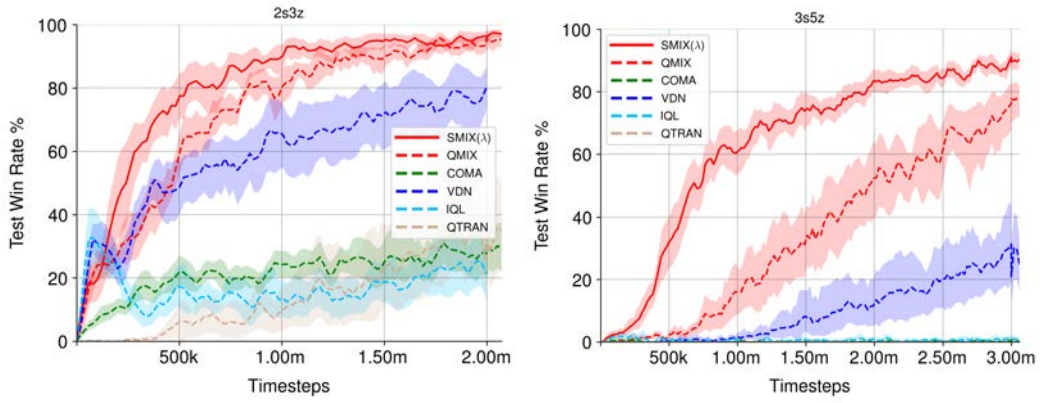
图 4.1　SMIX(λ) 以及对比方法 (QMIX, COMA, VDN, IQL, QTRAN) 在同质且数量对等地图 (3m 和 8m) 中的测试胜率比较。每个算法独立地进行 10 次性能评估，图中表示了平均性能以及其 95% 的置信区间 (阴影部分)。

4.5.3 实验结果

目前 QMIX[41] 和 COMA[40] 在星际争霸基准平台中目前表现最佳，因此本文中将这两种算法作为最先进的算法，VDN[48]、IQL[69] 以及 QTRAN[49] 作为比较的基准算法。

3m 及 8m. 在图4.1中，本文绘制了所有的比较算法在 3m 和 8m 中的测试结果。3m 和 8m 属于同种且数量对称的对战场景，由于其对战单元相对简单且对称，因此这两个地图为星际争霸平台中相对简单的情形，一般作为算法调试的测试地图。

从图4.1(a) 中可以发现，在 3m 中，除了 QTRAN，几乎所有的算法都可以在一定时间步中获得较好的结果，最终性能均可以收敛到接近 100% 的胜率。然而在 8m 中 (图4.1(b))，IQL 这种完全独立学习的算法在性能上的劣势表现出来，在最终性能上均明显低于其他算法，主要原因是该算法中的每个智能体将其他智能体视为环境的一部分，而其他智能体的策略也在不断改变之中，因此智能体的策略受环境非静态因素的影响，导致难以收敛。此外，该算法的奖励分配机制也是影响其性能的潜在因素之一。而 QTRAN 由于其优化的是联合动作空间，造成其优化代价过大，因此其在星际争霸这种复杂的环境中无法具有较好的表现。无论怎样，本文提出的算法 SMIX(λ) 就学习速度以及最终性能上均可以与其他算法比肩，在 3m 和 8m 之类的同种且对称的场景中，COMA 仅比 SMIX(λ) 快一点，但在最终性能方面却不如 SMIX(λ)。但实际上，由于地图相对容易，因此 SMIX(λ) 未能表现出较大的性能优势。



(a) 2s3z　　(b) 3s5z

图 4.2　SMIX(λ) 以及对比方法 (QMIX, COMA, VDN, IQL, QTRAN) 在异质且数量对等地图 (2s3z 和 3s5z) 中的测试胜率比较。每个算法独立地进行 10 次性能评估，图中表示了平均性能以及其 95% 的置信区间 (阴影部分)。

2s3z 和 3s5z. 在图4.2中，本文绘制了不同算法在异种类型地图 (2s3z 和 3s5z) 中的性能表现，相对于 3m 和 8m，这些地图在智能体数量和类型组合上更加复杂，两个地图均由狂热者 (Zealot) 和追踪者 (Stalker) 构成，主要差异在于数量上的不同。在图4.2(a) 中可以发现，在 2s3z 地图中，

SMIX(λ) 的学习速度快于 QMIX 近 2 倍，SMIX(λ) 在 1 百万个时间步时的便已经收敛，QMIX 几乎需要 2 百万个时间步。而将 SMIX(λ) 与 VDN 进行比较可以发现，VDN 就学习速度和最终效果上均劣于本文提出的方法，VDN 在训练 2 百万个时间步后其胜率只达到了 80%。IQL、COMA 以及 QTRAN 的性能则相应表现不佳，这三个算法的最终性能只达到了 30% 左右，远远逊色于本文提出的方法。

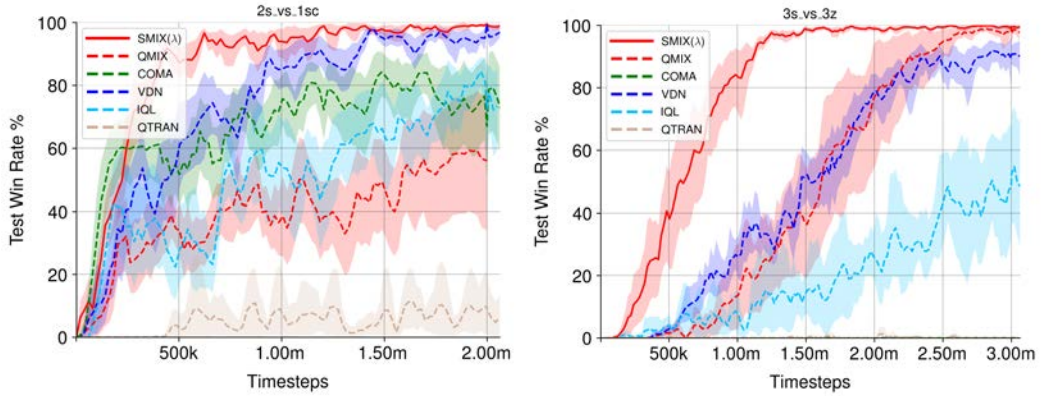
而在 3s5z 这种从数量上相对 2s3z 更复杂的地图中，智能体数量从 5 个增加到了 8 个，此时相对于 2s3z 地图来说，所有算法达到最高性能所需时间步数多于对应算法在 2s3z 中的表现。例如 SMIX(λ) 在 3s5z 中需要 3 百万个时间步才可以达到 90% 的胜率，而在 2s3z 中只需要 1 百万个时间步。而从不同算法之间的对比上来看，SMIX(λ)(红色实线) 在训练三百万个时间步后最终性能可以收敛到几乎 90% 的胜率，QMIX(红色虚线) 在此时却只能达到 75% 左右的胜率，其他方法则表现不理想，IQL(绿色虚线) 和 COMA(浅蓝色虚线) 以及 QTRAN(浅灰色虚线) 在该场景下甚至几乎无效。相较于在 3m 和 8m 中 SMIX(λ) 与其他算法的性能差距，在复杂场景中，SMIX(λ) 解决复杂多智能体协作的能力进一步体现出来。

3s_vs_3z 和 2s_vs_1sc. 上述 2s3z 和 3s5z 属于异质地图类型，本文在图4.3中绘制了所有算法在 3s_vs_3z 和 2s_vs_1sc 中的性能，这两种地图考验算法能否学会复杂的微技能。在 2s_vs_1sc 中，SMIX(λ) 需要不到 QMIX 和其他比较方法样本数量的一半就能达到渐近性能。在 3s_vs_3z 地图中可以看到最大的性能差距 (图4.3(b))。QMIX 需要接受将近 3 百万个时间步训练才能达到 100% 的测试获胜率，而一半的时间步足以使 SMIX(λ) 达到相同的获胜率。在 2s_vs_1sc 中，可以发现一个有趣的结果，即 VDN 可以实现比 QMIX 更好的性能 (请参见图4.3(a)) 。这表明，更简单的网络结构也可以具有足够的表示能力，而 VDN 优异性能的原因在于更简单网络结构的训练只需要相对较少的样本。COMA 在该场景下几乎无法学习到任何有用的策略，这可能是因为 COMA 的集中价值函数估计方法极大限制了算法的表现。总体而言，使用 λ-回报的 SMIX(λ) 相对于仅使用单步估计的 QMIX 具有明显的优势。

4.5.4 游戏视频回放分析

为了观察算法训练的智能体是否学习到了复杂的行为策略，本文将实际训练的模型在不同时刻的参数保存下来，训练完成后，加载不同时刻模型参数与游戏内置 AI 作战，以视频回放的方式观察智能体在不同训练阶段习得的作战行为。

本文以 3s5z 地图为例进行算法学得行为的分析。分析之前，首先详细介绍相关对战单元的基本属性。3s5z 地图中，对战双方均由 3 个追猎者 (Stalker) 和 5 个狂热者 (Zealot) 组合而成，单元基本信息见图4.4。追踪者的名字暗示了它打了就跑 (hit-and-run) 的能力，追踪者有相对较高的移动速度，并且可以远距离攻击，因此对于追踪者，一个较好的策略为“打了就跑”。对于狂热者，由于其移动速度较慢，并且只能够进行近身攻击，因此它在“风筝战术”面前也显得相



(a) 2s_vs_1sc　　(b) 3s_vs_3z

图 4.3　SMIX(λ) 以及对比方法 (QMIX, COMA, VDN, IQL, QTRAN) 在微操地图 (2s_vs_1sc 和 3s_vs_3z) 中的测试胜率比较。每个算法独立地进行 10 次性能评估，图中表示了平均性能以及其 95% 的置信区间 (阴影部分)。

(a) 追猎者 3D 模型及其基本信息　　(b) 狂热者 3D 模型及其基本信息

图 4.4　追猎者 (Stalker) 和狂热者 (Zealot) 的 3D 模型和基本信息。

当被动[1]，所以对付狂热者较好的策略便为使用“风筝战术”，通过追踪者较快的移动速度制约狂热者，同时使得自身不会收到伤害。

本文分别保存了 SMIX(λ) 使用随机策略 (未进行任何训练，图4.5)，训练 1 百万时间步 (图4.6) 以及训练 3 百万时间步 (图4.7) 后的模型参数[2]。

图4.5中展示了当 SMIX(λ) 未经任何训练的作战情况。从图4.5(a) 中可以看到，敌方单元几乎可以无任何单元伤亡的情况下获胜，SMIX(λ) 在五轮对战中全负，并且在每一轮对战中几乎无法对敌方单元造成大规模伤害，此时内置游戏 AI 使用固定的策略可以达到全胜。从图4.5(b) 及 (c) 中可以分析出随机策略无法取胜的原因：1) 在图4.5(b) 中我方追踪者的站位非常集中，在

[1]" 风筝战术" 即通过自己的射程和机动性优势来与火力强大的敌人周旋，最后将其击败。

[2]SMIX(λ) 在不同训练阶段对战的完整视频见：https://youtu.be/mY13RDVy-qc

(a) 敌方单元 (红) 几乎无需损失任何单元即可取胜。

(b) 我方追踪者 (绿) 站位非常集中，没有利用追踪者移动速度较快的优势。

(c) 敌方追踪者收到很好的保护，我方无法对其造成任何打击。

(d) 5 轮的对战中，我方单元共计获胜 0 次，胜率为 0%

图 4.5　3s5z 地图中，SMIX(λ) 使用随机策略 (无任何训练) 的行为以及测试胜率。绿色单元使用 SMIX(λ) 训练，红色单元为使用游戏内置 AI 的固定策略。

整个对战中几乎没有利用其速度优势进行分散站位以及使用“打了就跑”的策略。同时，集中站位容易导致对方狂热者无需大幅度移动即可对我方造成伤害。2) 图4.5(c) 中则显示出内置 AI 策略认为追踪者作为一个移动速度快、数量稀少且可远距离攻击的兵种，值得以牺牲狂热者来保护追踪者。此策略导致最后敌方具有足够数量的狂热者对我方造成持续伤害。

而在训练了 1 百万个时间步后，从图4.6中可以观察到 SMIX(λ) 学到了以下策略：

1. 围攻敌方狂热者。从图4.6(a) 中可以发现我方狂热者学会围攻敌方的狂热者，采取前后包夹的方式对敌方狂热者进行攻击。
2. 保护自身追踪者。从图4.6(b) 中可以发现当敌方狂热者攻击我方追踪者时，我方狂热者采取了保护自身追踪者的行为，这表明 SMIX(λ) 训练的策略能够挖掘到追踪者对于取胜的关键性。
3. 率先攻击对方追踪者。从图4.6(c) 中可以发现我方狂热者还可以学会先攻击敌方关键兵种的策略，这对于整体对战的取胜至关重要。否则敌方追踪者将对我方持续造成伤害。

训练了 1 百万个时间步后，SMIX(λ) 训练的策略胜率可达 40%。尽管 SMIX(λ) 可以学习到一些有效的策略，例如围攻，保护自身关键兵种以及优先攻击敌方关键兵种，但这些策略并没有利

用关键兵种的特性 (例如移动速度快) 来对敌方造成充分的伤害。

(a) SMIX(λ) 训练的策略学会了围攻狂热者的技能。

(b) 我方狂热者保护追踪者，以防止受到敌方狂热者的攻击。

(c) 我方狂热者学会先去攻击敌方追踪者，防止其对我方造成远距离伤害。

(d) 5 轮的对战中，我方单元共计获胜 2 次，胜率为 40%。

图 4.6　3s5z 地图中，SMIX(λ) 在训练 1 百万个时间步后的学得的技能以及测试胜率。绿色单元使用 SMIX(λ) 训练，红色单元为使用游戏内置 AI 的固定策略。

在训练了 3 百万个时间步后，SMIX(λ) 学到了更加复杂的协作策略，如图4.7所示。

1. 追踪者学会了打了就跑的策略。图4.7(a) 显示敌方狂热者一直希望近距离靠近我方追踪者，以对我方狂热者采取近身伤害，但我方追踪者一直采取后退的策略，与敌人保持一定的距离，此即“打了就跑”的策略，充分利用自身移动速度较快的优势制约对手。
2. 追踪者学会了分散站位。图4.7(b) 显示出我方追踪者学会了分散站位。追踪者作为较为稀缺且可远距离攻击敌方的兵种，可以保证敌方狂热者无法通过较少的移动对多个追踪者造成伤害, 同时只有分散站位才可以保证不会受到敌方团灭。
3. 狂热者学会了围攻敌方追踪者。图4.7(c) 显示出我方狂热者此时转变之前围攻敌方狂热者的策略，转而围攻敌方追踪者。这体现出 SMIX(λ) 能够学习到追踪者的关键作用，通过优先消灭敌方关键兵种来保证自身受到较少伤害。

此时，SMIX(λ) 训练的策略胜率达到了 80%。可见 SMIX(λ) 可以通过一定时间的训练可以学到复杂的协作策略。

在图4.8中，本文给出了 VDN，QMIX 以及 SMIX(λ) 在均训练 1 百万时间步的条件下在 5

(a) SMIX(λ) 训练的策略学会分散站位，防止 Zealots 进行近身攻击。

(b) 我方追踪者学会以分散站位的方式围攻敌方追踪者。

(c) 我方狂热者学会以围攻的方式防止敌方追踪者快速移动。

(d) 5 轮的对战中，我方单元共计获胜 4 次，胜率为 80%

图 4.7　3s5z 地图中，SMIX(λ) 在训练 3 百万个时间步后的学得的技能以及测试胜率。绿色单元使用 SMIX(λ) 训练，红色单元为使用游戏内置 AI 的固定策略。

局对战中的胜负情况。SMIX(λ) 以五局三胜的战绩高于 VDN 以及 QMIX[1]。

4.5.5　SMIX(λ) 通用性分析

SMIX(λ) 关心使用异策略数据结合 λ-回报的方式来进行集中价值函数的估计。这种方法实际上也可以应用到其他基于值函数估计的多智能体强化学习算法中。

为了说明本文提出的集中价值函数估计的优越性，将 SMIX(λ) 中使用的价值函数估计方法应用到下列算法中：COMA，VDN, IQL 以及 QTRAN。通过将这些算法原始的集中价值函数估计策略替换为本文提出的方法，可获得四个新的算法，本文将它们分别称为：SMIX(λ)-COMA, SMIX(λ)-VDN，SMIX(λ)-IQL 以及 SMIX(λ)-QTRAN。图4.9展示了不同算法在 6 个不同地图中的性能。

总的来说，在加入本文提出的集中价值函数估计方法之后，原有算法的性能就学习速度以及最终性能而言，都得到了不同程度的提升。

SMIX(λ)-VDN 极大地提高了 VDN 的性能。特别是在困难的 3s5z 场景中，SMIX(λ)-VDN

[1]完整视频见：https://youtu.be/2Eut_y-9neQ。

图 4.8　3s5z 地图中，VDN，QMIX 以及 SMIX(λ) 均训练 1 百万时间步后的测试胜率。

达到了大约 75% 的最终获胜率，为 VDN 的两倍多 (近 30%)。这种性能上的提高可归功于 λ-回报以及学习过程中无需依赖不切实际的集中贪婪假设。

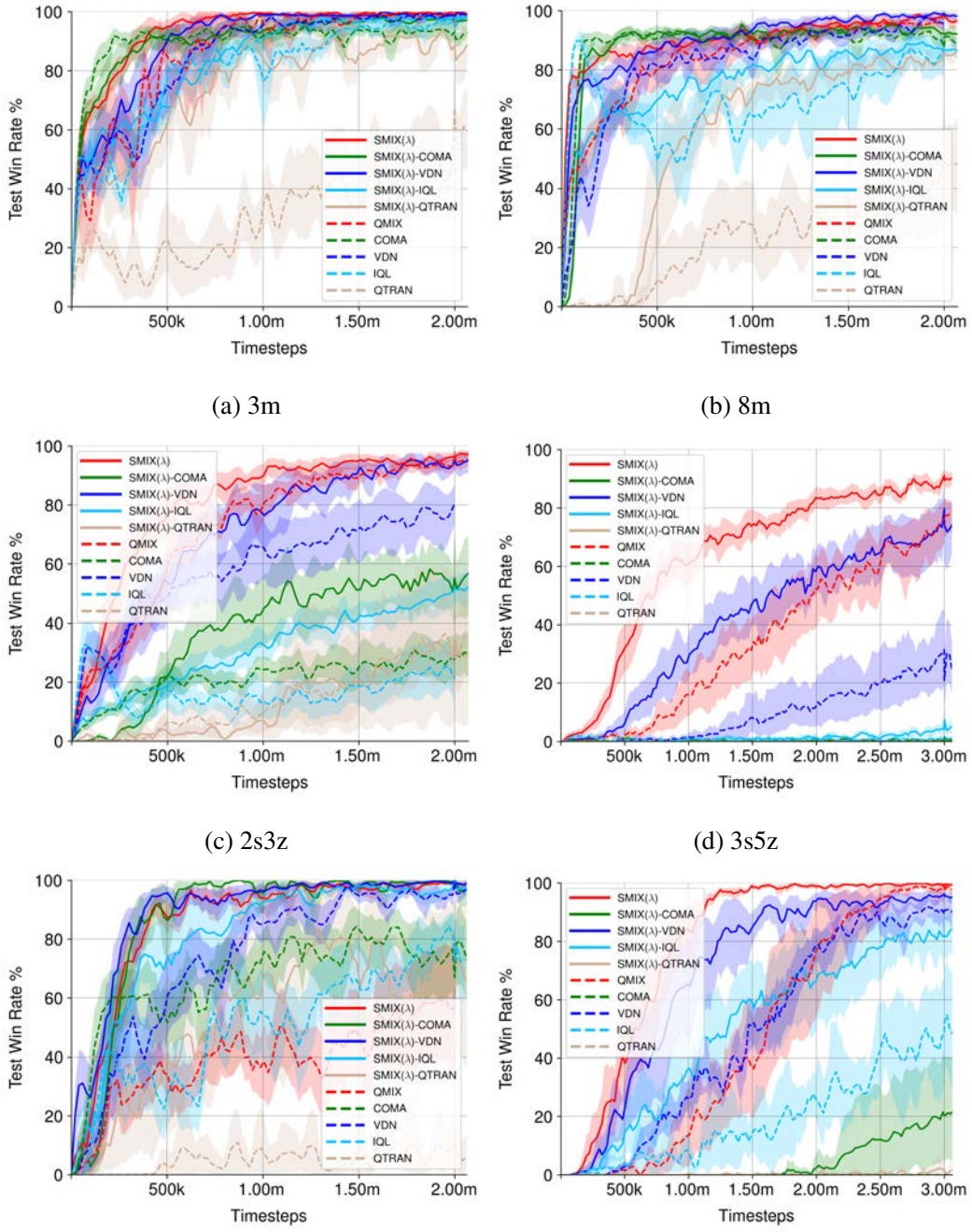
此外，还可以发现 SMIX(λ)-VDN 在大多数情况下的性能甚至比 QMIX 更好。注意，VDN 使用分散 Q 值的线性组合 (本文的 SMIX(λ)-VDN 也是如此)，而 QMIX 通过以非线性的方式结合分散 Q 值对 VDN 进行了拓展，这使得 QMIX 可以表示更加复杂的集中价值函数。然而，本文的结果表明，VDN 的性能瓶颈可能并不是 VDN 比较有限的表示能力，而是如何有效地平衡集中价值函数估计中的偏差和方差。

类似的性能改进也可以在 COMA 中看到，这可以归功于成功地利用了异策略数据。因为 COMA 也采用了 λ-回报，但只使用同策略数据。

另一个发现是，本文的方法也适用于 IQL，它是一种完全分散的多智能体强化学习算法。这表明本文的方法不仅适用于集中价值函数估计，也适用于分散情况。类似的，将本文提出的值函数估计方案加入到 QTRAN 中，在 3m、8m 以及 2s_vs_1sc 中均看到了非常明显的性能提升。

值得一提的是，如果原始方法不起作用，比如 COMA、IQL、QTRAN 以及它们的对应方法在 3s5z 中不起作用，那么扩展的方法可能不会得到改进 (图4.9d)。原因可能是约束 COMA、IQL 和 QTRAN 在 3s5z 中性能的主要瓶颈并不在于值函数估计不准确，而在于其他诸如智能体数量多、多智能体信用分配以及优化方法等问题。

在表4.3和表4.4中，本文给出了不同算法在训练一百万个时间步之后的量化性能。可以看出，就样本效率而言，本文的方法几乎在所有地图中均可以比肩甚至优于其对应算法。



(a) 3m (b) 8m

(c) 2s3z (d) 3s5z

(e) 2s_vs_1sc (f) 3s_vs_3z

图 4.9　将 SMIX(λ) 的价值函数估计方法应用到其他方法中 (QMIX, COMA, VDN, IQL, QTRAN) 后，相应算法在 6 个微操地图中的测试胜率比较。本文的方法及其对比方法分别用同种颜色的实线和虚线表示。每个算法独立地进行 10 次性能评估，图中表示了平均性能以及其 95% 的置信区间 (阴影部分)。

表 4.3　训练一百万个时间步后不同算法测试胜率在 3m, 8m 以及 2s3z 地图中的均值，标准差以及中位数。性能最高的结果的字体进行了加粗。

算法	3m		8m		2s3z	
	mean ± *std*	median	mean ± *std*	median	mean ± *std*	median
SMIX(λ)	**99** (±0)	**99**	**91** (±3)	**90**	**90** (±4)	**91**
QMIX	95 (±3)	95	90 (±3)	89	81 (±7)	81
SMIX(λ)-COMA	**93** (±8)	**97**	**92** (±2)	**93**	**44** (±18)	**47**
COMA	92 (±2)	93	90 (±2)	91	24 (±6)	24
SMIX(λ)-VDN	**98** (±0)	**98**	**94** (±3)	**93**	**78** (±14)	**79**
VDN	95 (±2)	95	86 (±5)	87	64 (±16)	71
SMIX(λ)-IQL	**91** (±4)	**94**	**80** (±5)	**79**	**32** (±8)	**31**
IQL	83 (±9)	86	59 (±15)	58	14 (±10)	13
SMIX(λ)-QTRAN	**84** (±10)	86	**72** (±6)	72	5 (±8)	2
QTRAN	29 (±11)	29	24 (±17)	21	**9** (±10)	**5**

4.5.6　消融分析

在本节中，给出 SMIX(λ) 中不同组成部分的消融分析。具体地，分别研究使用异策略数据对性能的影响，以及平衡方差和偏差的必要性。

4.5.6.1　异策略数据的影响

异策略数据的多少通过调整缓冲池大小 b 得以实现，由于实现中使用固定的缓冲池大小，并且每次采样到新数据之后，将该数据覆盖缓冲池中最旧的数据，因此 b 越小，则用来更新策略的数据越接近同策略数据，b 越大则越接近异策略数据。同策略的 SMIX(λ) 对应于 $b = 4$，异策略 SMIX(λ) 为其中 $b > 4$ 的变体。异策略数据随着 b 的增大而增多。本文在图4.10中给出了使用不同 b 取值后 SMIX(λ) 的性能表现。

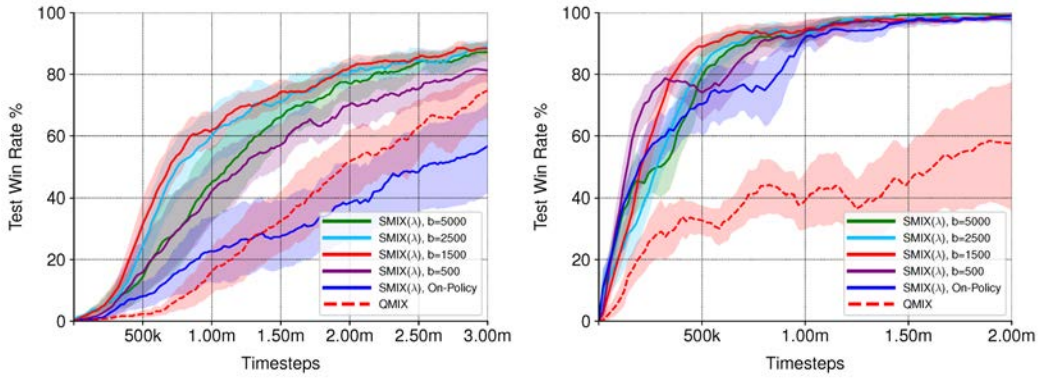
如图4.10a和4.10b所示，在所选的场景中，所有包含异策略数据 ($b > 4$) 的 SMIX(λ) 的性能都优于其同策略版本 ($b = 4$)。值得注意的是，在 $b = 1500$ 的情况下，无论是就最终胜率还是学习速度来说，SMIX(λ) 的性能几乎都是 3s5z 中同策略版本算法的两倍。

然而，更多的异策略数据并不总是会带来更好的性能，一旦缓冲区大小超过某个阈值，性能甚至可能会下降。这是因为缓冲区大小对应于定理2中的 ϵ，它度量目标策略 $\boldsymbol{\pi}$ 和行为策略 $\boldsymbol{\mu}$ 之间的差异。缓冲区较小使得 SMIX(λ) 的样本效率更低，但一个更大的缓冲区则会导致一个更

表 4.4　训练一百万个时间步后不同算法测试胜率在 3s5z, 2s_vs_1sc 以及 3s_vs_3z 地图中的均值，标准差以及中位数。性能最高的结果的字体进行了加粗。

算法	3s5z		2s_vs_1sc		3s_vs_3z	
	mean ± *std*	median	mean ± *std*	median	mean ± *std*	median
SMIX(λ)	**61** (±11)	**62**	**94** (±5)	**96**	**84** (±14)	**88**
QMIX	16 (±12)	11	39 (±19)	45	15 (±20)	9
SMIX(λ)-COMA	**0** (±0)	**0**	**97** (±4)	**100**	**0** (±0)	**0**
COMA	**0** (±0)	**0**	77 (±11)	78	**0** (±0)	**0**
SMIX(λ)-VDN	**29** (±12)	**26**	**96** (±2)	**97**	**67** (±25)	**83**
VDN	1 (±2)	0	86 (±8)	88	27 (±9)	27
SMIX(λ)-IQL	**0** (±0)	**0**	**92** (±6)	**94**	**35** (±21)	**31**
IQL	**0** (±0)	**0**	51 (±22)	54	5 (±4)	6
SMIX(λ)-QTRAN	**0** (±0)	**0**	**63** (±24)	**70**	**0** (±0)	0
QTRAN	**0** (±0)	**0**	**6** (±11)	**0**	**0** (±0)	0

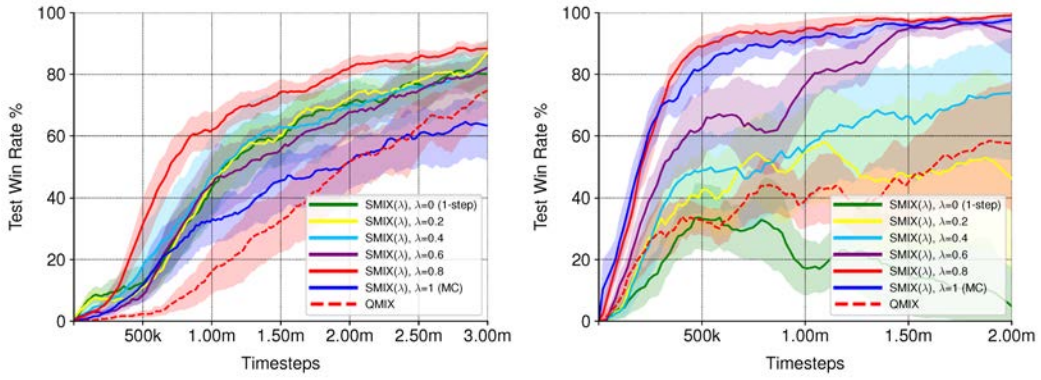
宽松的误差界，使得集中价值函数估计具有较大的偏差。在实践中，适当的缓冲区大小 $b = 1500$ 是一个较好的折中选择。



(a) 在 3s5z 地图中，SMIX(λ) 使用不同缓冲池大小 b 的性能。

(b) 在 2s_vs_1sc 地图中，SMIX(λ) 使用不同缓冲池大小 b 的性能。

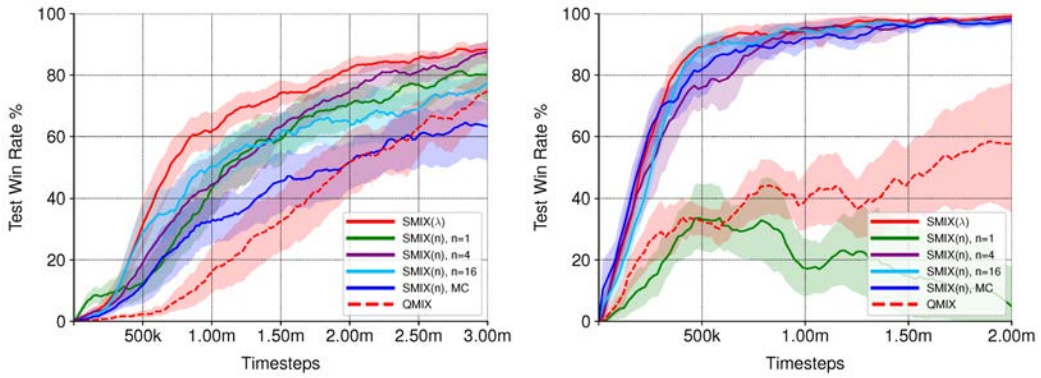
图 4.10　在两个不同的场景中，异策略数据的多少对 SMIX(λ) 性能的影响。每个算法独立地进行 10 次性能评估，图中表示了平均性能及其 95% 的置信区间 (阴影部分)。



(a) 在 3s5z 地图中，SMIX(λ) 使用不同的 λ 取值的算法性能。

(b) 在 2s_vs_1sc 地图中，SMIX(λ) 使用不同的 λ 取值的算法性能。

图 4.11　在两个不同的场景中，λ 取值的多少对 SMIX(λ) 性能的影响。每个算法独立地进行 10 次性能评估，图中表示了平均性能及其 95% 的置信区间 (阴影部分)。



(a) 3s5z 地图中，SMIX(n) 使用不同的 n 取值的性能。

(b) 2s_vs_1sc 地图中，SMIX(n) 使用不同的 n 取值的性能。

图 4.12　在两个不同的场景中，多步值函数评估的步数 n 对 SMIX(λ) 性能的影响。每个算法独立地进行 10 次性能评估，图中表示了平均性能及其 95% 的置信区间 (阴影部分)。

4.5.6.2　λ-回报和多步回报

为研究多智能体值函数估计问题中平衡偏差和方差的必要性，本文调整了参数 λ 以探究偏差和方差的影响。其中较大的 λ 表示较小的偏差，相应的方差较大；而较小的 λ 则相对应于较大的偏差和较小的方差。$\lambda = 0$ 和 $\lambda = 1$ 恰好对应了两种极端的情形。

- $\lambda = 0$ 时，SMIX(λ) 相当于单步回报 (对应于最大偏差和最小方差);
- $\lambda = 1$ 时，SMIX(λ) 等价于蒙特卡洛 (MC) 回报 (∞ 步，对应于最小偏差和最大方差)。

如图4.10a和4.10b所示，SMIX(λ) 在 $\lambda = 0.8$ 时，可在选定的地图中一致地获得最佳性能。

$\lambda = 1$ (MC 回报，蓝线) 的方法在 3s5z 中表现最差，而在 2s_vs_1sc 中表现较好。$\lambda = 0$ (单步回报，绿线) 则恰恰相反，其在 2s_vs_1sc 中表现最差，在 3s5z 中表现却较好。这样的结果表明不同的对战地图对算法方差和偏差的影响是有不同侧重的，2s_vs_1sc 中偏差占了主导因素，因此令 $\lambda = 1$ 可以具有较好的性能，而 3s5z 中方差则为影响算法性能的主导因素。但无论如何，$\lambda = 0.8$ 都可以一致地表现得最好，这恰恰说明了平衡算法值函数估计中的方差和偏差的重要性。

除了改变 λ 取值，本文还评估了 SMIX(λ) 的一个变种——SMIX(n)，SMIX(n) 使用 n 步回报代替 λ-回报来作为 TD 目标，也就是说:

$$y_t^{tot} = \sum_{i=1}^{n} \gamma^{i-1} r_{t+i} + \gamma^n Q(\boldsymbol{\tau}_{t+n}, \mathbf{a}_{t+n}; \theta^-). \tag{4.13}$$

在 SMIX(n) 中也可以看到与 SMIX(λ) 改变 λ 类似的结果 (图4.12a和4.12b)。其中 $n = 4$ 的 SMIX(n) 在 3s5z 中表现最佳，而 $n = 16$ 的 SMIX(n) 在 2s_vs_1sc 中表现最好。对于 SMIX(n)，很难找到与 SMIX(λ) 一样在所有地图中均表现良好的 n 值。总而言之，有必要在多智能体问题中平衡偏差和方差，而 λ-回报则可以作为实现这种权衡的一种非常便捷的方法。

4.5.7 可拓展性

本小结探究 SMIX(λ) 算法的可拓展性。即随着智能体数量的增加，相应算法性能的变化。

表4.5中的结果显示了训练一百万步后 SMIX(λ) 和 QMIX 的可拓展性比较。总体而言，两种方法的性能都会随着智能体数量的增加而降低。但是，本文的 SMIX(λ) 仍然优于 QMIX，尤其是在困难的地图中 (例如 25m)。具体来说，在 3 个智能体的地图中 (3m 地图)，SMIX(λ) 以 99% 的获胜率在比较的方法中获得最佳性能。通过将智能体数量增加到 8 个 (8m 地图) 时，由于任务的挑战性更高，因此所有方法的性能都会下降，而本文的方法在比较的方法中仍然表现最好。最后，当智能体数量增加到 25(25m 地图) 时，QMIX 的性能急剧下降，而 SMIX(λ) 则只有轻微的降低。这些结果表明，与 QMIX 相比，SMIX(λ) 方法中使用的集中价值函数估计方法具有更好的可扩展性，并且在具有挑战性的任务中具有更强的鲁棒性。最后，值得一提的是，在 25m 地图中具有 25 个智能体，其联合作用空间将高达 $|A|^{25}$，这样大的联合动作空间对任何多智能体强化学习方法都构成了巨大的挑战。

4.6 本章小结

在“集中训练、分散执行”框架下的多智能体强化学习的主要挑战之一是如何更好地估计集中价值函数。但是，多智能体环境的稀疏经验和环境的非静态性一直是集中价值函数估计中具有挑战性的任务。为了解决这个问题，本文提出 SMIX(λ) 方法。实验结果表明，本文的方法通过以下三个方面来提高集中价值函数的准确性，从而大大改善了多智能体强化学习算法的性能：(1) 消除贪婪的假设以学习更灵活的函数结构，(2) 使用异策略学习，以减轻经验稀疏的问

表 4.5　SMIX(λ) 以及 QMIX 的可拓展性比较 (均训练 1 百万个时间步)。

算法		SMIX(λ)	QMIX
3m	mean ± *std*	**99** (±0)	95 (±3)
	median	**99**	95
8m	mean ± *std*	**91** (±3)	90 (±3)
	median	**90**	89
25m	mean ± *std*	**75** (±26)	30 (±17)
	median	**93**	24

题并增强探索，(3) 使用 λ-回报来处理环境的非马尔可夫性质，并平衡值函数估计中的偏差和方差。本文的结果还表明，通过用 SMIX(λ) 中所用值函数估计方法替换其它方法的集中价值函数估计方式，都可以使得原始方法获得很大的性能提升。最后，本文的分析表明，SMIX(λ) 尽管使用了无需重要性采样的异策略学习，但其仍然具有良好的收敛性保证，这在多智能体设置中是非常具有优势的。最后本文通过对异策略数据的消融分析以及 λ 参数的敏感性分析说明了使用异策略数据以及平衡方差偏差的有效性。

第五章　多智能体强化学习在广告自动竞价中的应用

本章介绍本文的第二个主要工作，即提出了用以解决广告自动竞价问题的多智能体框架MAAB (Multi-Agent Auto-bidding)。

5.1　简介

在自动竞价问题中，广告平台为广告主设计自动出价智能体，每个智能体都为每一类别的广告主学习一种特定的出价策略，以在一定约束下优化广告主期望的目标。例如，在一般的广告平台中，存在三种典型的自动出价智能体：CLICK 智能体，CONV 智能体和 CART 智能体，它们分别在预算约束下优化点击次数，购买转化次数以及添加到购物车的数量。这些自动出价智能体通过竞标每个广告展现机会来相互竞争。在每一次广告展现机会到来后，平台将为参与的自动出价智能体发起广告拍卖，每次拍卖中，自动竞价智能体都根据广告主的目标、约束以及展现价值预估值（例如，pCTR，pCVR 或 pCART）为这个展现机会出价。在收到所有智能体的出价之后，拍卖机制决定获胜的广告主和相应的展现价格。自动竞价的具体过程如图5.1所示。

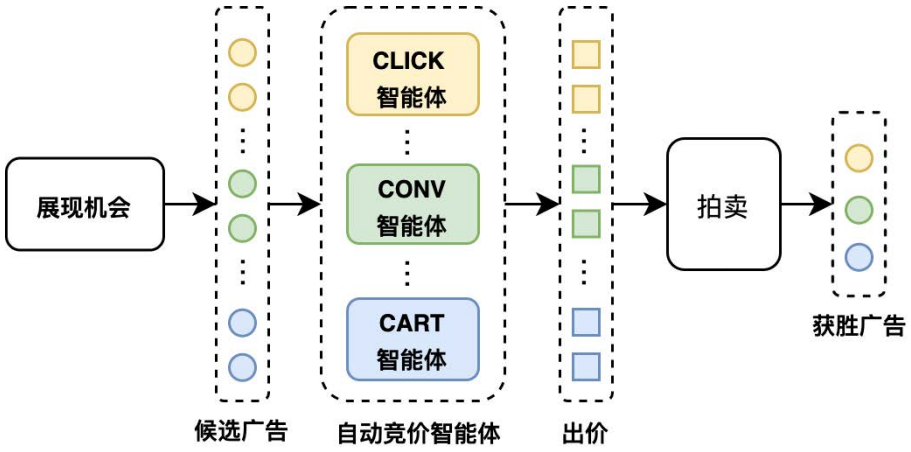


图 5.1　自动竞价过程示意图。

为了学习自动竞价智能体的出价策略，实时竞价 (Real-Time Bidding, RTB)[26,85,86] 研究中常常将每种自动竞价智能体形式化为解决一个独立的优化问题，而将其他智能体隐式地编码为拍卖环境的一部分。然而，这种建模方案忽略了拍卖本质上是一种分布式多智能体系统的事实，因为拍卖的结果在很大程度上取决于所有自动竞价智能体的竞价行为。此外，一方竞价智能体的策略行为改变会影响其他智能体的策略，反之亦然。

另一种方案为基于多智能体框架学习所有自动竞价智能体的出价策略。但是在此框架下设计自动竞价智能体的出价策略依然存在一些问题。首先，自动竞价智能体之间复杂的竞争合作关系阻碍了广告平台协同优化单个智能体的效用和系统整体性能。一方面，在完全竞争的多智能体范式中，每个广告主的效用可以被极大地优化。例如，预算充足的自动出价智能体会出于自身利益而贪婪地出高价并赢得大量广告展现机会。但从全局的视角来看，这种分配方案会导致社会福利较低。另一方面，完全协作的多智能体范式可以解决此问题，因为所有自动竞价智能体都旨在实现社会福利最大化。但是，这样做可能会牺牲一些广告主的效用以获得较大的社会福利，这对于在线广告生态的长期繁荣而言并不健康。

一个更合适的方法是建立一种混合协作竞争 (Mixed Cooperative-Competitive, MCC) 的框架，使得平台能够在个体效用和系统性能之间做出灵活的权衡。目前已有方法是通过手动修改奖励函数[69] 或更改与环境相关的参数[47,70] 来达到该目标。但是在竞价拍卖中，仍没有具体的奖励函数可以为自动竞价智能体之间建立一种混合协作竞争关系。此外，环境相关的参数通常只能在模拟器中进行控制[70]，而现实世界的环境中往往不具备这样的条件。

然而，混合协作竞争框架中的协作将不可避免地降低平台的收入。这是因为以合作为基础的自动竞价智能体将会与其他智能体共谋，来为每一次展现机会出低价[87] 以预留尽可能多的预算。我们可以使用最优拍卖[88] 中的经典方法——保留价[60,61] 来提升平台的收入。然而，如何在混合协作竞争的框架下，设置一个既可以保证平台收入又可以最大化社会福利的保留价仍是一个开放性问题。

联合考虑上述设计上的挑战，本文针对自动竞价问题提出了一种混合协作竞争的多智能体强化学习框架，称为 MAAB。首先，为了实现自动竞价智能体之间协作与竞争关系的折中，本文提出了一种基于温度调控的奖励分配方法，通过一种基于 softmax 的方式将总体奖励按与智能体出价正比的方式进行分配，总而建立了自动竞价智能体之间的混合协作竞争关系。同时，softmax 函数中的温度调控参数可以作为一种用以调控智能体之间协作性与竞争性的工具。其次，受最优拍卖中[60–62] 的保留价启发，本文设计门槛智能体来为每一个自动竞价智能体学习一个具有个性化的竞价门槛，以减少平台收入的降低。直觉上，门槛智能体的目标是提高竞价门槛以提高平台的收入，然而自动竞价智能体则具有一个相反的目标，即降低竞价门槛使得可以以较低的价格赢得广告展现机会。门槛智能体是以一种对抗的方式与自动竞价智能体进行联合训练，直到双方的策略达到某种均衡点。

本文在两个智能体的自动竞价环境中对本文所提方法进行验证，实验表明本文方法可以在提高社会福利的同时，减少平台收入的降低。

5.2 背景

5.2.1 自动竞价模型

在自动竞价服务中，广告主希望可以在固定的预算下最大化其累积展现价值[25]。本文考虑预算约束下的自动竞价问题。单个广告主的自动竞价过程如下所述。在一个时间段内（例如一天），有 T 个展现机会按序而来，给定广告主的预算 B^i，自动竞价智能体实时地为广告主 i 自动出价 b_t^i，如果拍卖中的最高价格是 b_t^i，则该广告主可以展示其广告并将 v_t^i 作为其展现价值。赢得一个展现机会的同时还需要付出该次展现的相应成本 c_t，该成本在二价拍卖中由第二高的出价所决定。如果总成本达到预算限制或没有剩余的展现机会了，则该广告主的竞价过程终止。为广告主 i 出价的自动竞价智能体的目标是在预算约束下最大化累积的展现价值：

$$\max \sum_{t=1}^{T} v_t^i \cdot x_t^i \tag{5.1}$$

$$s.t. \sum_{t=1}^{T} c_t \cdot x_t^i \leq B^i, \tag{5.2}$$

其中 $x_t^i \in \{0,1\}$ 表示广告主 i 是否赢得展现 t。在以下章节中，如果没有特别说明，本文也使用 i 表示为广告主 i 进行自动竞价的智能体 i。

5.2.2 马尔可夫博弈

由 n 个智能体构成的部分观测马尔可夫博弈[89] 可使用元组 $< \mathcal{S}, P, \{Z^i, O^i, \mathcal{A}^i, r^i\}_{i=1}^n, \gamma >$ 进行描述。$s \in \mathcal{S}$ 为环境的真实状态。在每一时间步，每一个智能体从观测函数 $Z^i : \mathcal{S} \rightarrow O^i$ 中得到其观测值 o^i，然后基于其策略 π^i 采取一个动作 $a^i = \pi^i(o^i)$。在所有智能体采取联合动作 $\mathbf{a} = (a^1, \cdots, a^n)$ 之后，每一个智能体获得一个标量奖励 $r^i : \mathcal{S} \times \mathcal{A}^1 \times \cdots \times \mathcal{A}^n \rightarrow \mathbb{R}$, 同时环境依据状态转移函数 $P(s'|s, \mathbf{a}) : \mathcal{S} \times \mathcal{A}^1 \times \cdots \times \mathcal{A}^n \rightarrow \mathcal{S}$ 转移到下一个状态 s'。$\gamma \in (0,1]$ 是奖励折扣因子。每一个智能体旨在学习其策略以最大化期望回报 $R^i = \mathbb{E}[\sum_t \gamma^t r_t^i]$。

本文使用马尔可夫博弈来建模自动竞价的多智能体系统。具体的建模如下所述。在每一时间步 t, 自动竞价智能体 i 根据观测 $o^i = (B_t^i, v_t^i, ts_t^i)$ 决定其出价 $b_t^i = \pi^i(o_t^i)$, 其中 B_t^i 为剩余预算，v_t^i 为展现价值，ts_t^i 为剩余的展现机会。智能体出价被裁剪至区间 $[b_{min}, b_{max}]$, 其中 b_{min} 和 b_{max} 分别为最小及最大的可行出价。每一个智能体的环境奖励为 $r_t^i = v_t^i \cdot x_t^i$, 成本 c_t 由第二高的出价所决定。之后环境转移到下一状态，每一个智能体的下一观测值为 $o_{t+1}^i = (B_t^i - c_t \cdot x_t^i, v_{t+1}^i, ts_t^i - 1)$。每一个自动竞价智能体的目标为最大化其期望累积竞得展现价值: $\max_{\pi^i} \mathbb{E}[\sum_t \gamma^t r_t^i]$。

5.2.3 独立学习

在马尔可夫博弈[89] 的框架下，最被广泛使用的多智能体强化学习方法是同时为每一个智能体学习其各自的策略或值函数[45,69]。独立 Q 学习[45,69] 使用 Q-learning 或 DQN[36] 分别为每一

个智能体学习一个 Q 函数, 其中所有智能体共享相同的环境并同时进行学习。为简洁起见，在以下章节中，本文将独立 Q 学习方法简称为 IL (Independent Learner)。

IL 使用一个参数为 θ 的深度神经网络表示每个智能体的动作值函数 $Q^i(o^i, a^i)$。每个 $Q^i(o^i, a^i)$ 的训练细节与 DQN[36] 相同。IL 也使用一个经验回放缓冲区 $\mathcal{D}$ 来存储所有智能体的转移元组 $\{(o^i, a^i, r^i, o^{i\prime})\}_{i=1}^n$，其中 $o^{i\prime}$ 为智能体 i 根据其观测 o^i 执行动作 a^i，并得到奖励 r^i 后的下一个观测。θ^i 通过从经验回放缓冲区中采样并最小化以下损失函数来进行学习:

$$\mathcal{L}(\theta^i) = \mathbb{E}_{(o^i, a^i, r^i, o^{i\prime})\sim\mathcal{D}}\left[(y^i - Q^i(o^i, a^i; \theta^i))^2\right], \tag{5.3}$$

其中目标 $y^i = r^i_{\text{train}} + \gamma \max_{a^{i\prime}} Q(o^{i\prime}, a^{i\prime}; \hat{\theta}^i)$，$\hat{\theta}^i$ 为目标网络的参数，目标网络的参数每过固定的一段时间阶段性地更新为 θ^i。r^i_{train} 为直接用来训练参数 θ^i 的奖励信号。简单起见，本文给出不同奖励值的定义以便在后续章节中对本文所提方法进行描述:

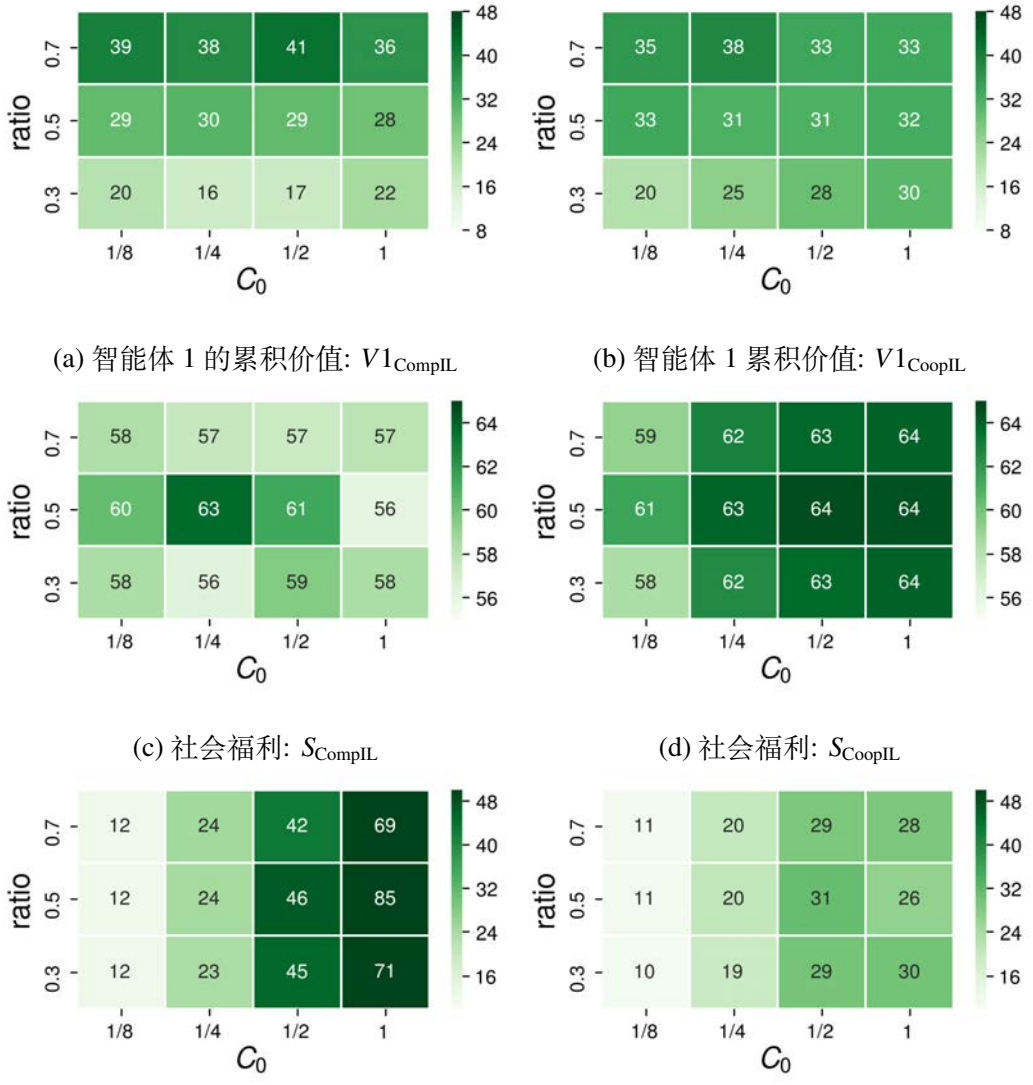
- **环境奖励** r^i 为环境返回给每一个智能体 i 的奖励值。
- **训练奖励** r^i_{train} 用以直接训练每一个智能体。本文称直接使用 $r^i_{\text{train}} = r^i, \forall i$ 来训练 IL 的方法为竞争的独立学习 (Competitive Independent Learner, CompIL) 方法。
- **总体奖励** $r_{\text{tot}} = \sum_{i=1}^n r^i$ 定义为所有智能体的环境奖励之和。总体奖励衡量了该次展现机会的分配结果对于社会福利的贡献。本文称使用 $r^i_{\text{train}} = r_{\text{tot}}, \forall i$ 作为奖励信号来训练每一个智能体的方法为协作的独立学习 (Cooperative Independent Learner, CoopIL) 方法。

本文还定义了合作和竞争关系。考虑一个有两个智能体的单轮拍卖，其归一化到相同的尺度后的展现价值分别为 v^1 和 v^2。假设 $v^1 > v^2$，如果他们的出价满足 $b^1 > b^2$，那么他们的关系是**合作**，否则是**竞争**。这一定义符合协作有助于获得更好的社会福利的直觉，因为社会福利的一个衡量标准是所有智能体的奖励值之和[90]。

5.3 独立学习的行为表现

在本节中，本文分析 CompIL 和 CoopIL 在自动竞价背景下分别展现出的行为。我们发现 CompIL 方法会导致寡头现象的产生，进而导致较差的社会福利。相反，CoopIL 获得了更好的社会福利，然而却以减少平台收益为代价。

为了说明这一点，本文设计了一个具有两个智能体的竞价环境，在该环境中，两个自动竞价智能体都基于一定的预算约束来竞价，以最大化各自竞得的累积展现价值。本文分别训练了 CompIL 和 CoopIL 约 50k 条轨迹，并根据以下三个评估指标比较它们的最终性能: 1) 智能体 1 的总价值 ($V1$)，2) 社会福利 (S)，以及 3) 平台收益 (P)。本文将社会福利定义为两个智能体价值之和，平台收益定义为 GSP 机制下的总成本。由于 $V2$ 与 $V1$ 基本对称，因此结果中并未显示。我们实验了预算约束参数 C_0 和预算比例参数 $ratio$ 在不同组合下两种方法各自的性能。较大的 C_0 对应较大的预算总额，而参数 $ratio$ 控制分配给智能体 1 的总预算的比例。两个智能体



(a) 智能体 1 的累积价值: $V1_{\text{CompIL}}$

(b) 智能体 1 累积价值: $V1_{\text{CoopIL}}$

(c) 社会福利: S_{CompIL}

(d) 社会福利: S_{CoopIL}

(e) 平台收入: P_{CompIL}

(f) 平台收入: P_{CoopIL}

图 5.2　CompIL 和 CoopIL 在不同预算约束参数 C_0 和预算比例参数 *ratio* 下的收敛性能。

的展现价值均从均值为 0.5，方差为 1 的正态分布中采样而得。小节5.2.2给出了建模细节，小节5.5给出了环境以及参数更多的细节。图5.2给出了实验结果，横坐标代表不同的 C_0，纵坐标表示不同的 *ratio* 值，每个彩色单元表示 CompIL 或 CoopIL 在不同的 C_0 和 *ratio* 的参数组合下 $V1$，S 或 P 的值。

图5.2a 显示了智能体 1 的累积价值。在 $ratio = 0.7$ 时，也即智能体 1 具有更多的预算，可以发现智能体 1 的累积价值为 $V1_{\text{CompIL}} = 39, 38, 41, 36$, 这显然远比智能体 2 的累积价值要高很多 (由于对称性质，智能体 2 的累积价值可以通过将 $V1_{\text{CompIL}}$ 中的 *ratio* 设置为 0.3 得到，也就是 20, 16, 17 和 22)。这表明智能体 1 通过出更高的价格赢得了大部分展现机会，智能体 1 也

因此成为**寡头**。本文将寡头定义为在具有足够预算的情况下通过贪婪地出高价而得到了大部分展现机会的智能体。寡头现象的产生导致了更差的社会福利。如图5.2c以及图5.2d所示, CompIL 就社会福利而言比 CoopIL 差很多，特别是在足够预算的情况下。例如当 $C_0 = 1$ 时，CoopIL 的社会福利远高于 CompIL (前者为 64，64，64，后者为 57，56，58)，这是因为当预算足够多时，CompIL 只是简单地给出一个非常高的价格以赢得所有展现机会，这会造成不适当的展现机会分配结果，也就是说，智能体 1 的展现机会价值低于智能体 2，然而智能体 1 却给出了一个较高的出价。

CoopIL 通过阻止寡头的产生来提升社会福利。这可以通过比较图5.2a和图5.2b中的 $V1_{\text{CoopIL}}$ 和 $V1_{\text{CompIL}}$ 看出：当被分配了更多的预算时 ($ratio = 0.7$)，智能体 1 的累积价值从 39, 38, 41, 36 降低到 35, 38, 33, 33；然而当被分配了较少的预算时，智能体 1 的累积价值从 20, 16, 17, 22 提高到了 20, 25, 28, 30。这表明 CoopIL 更多的是依据两个智能体的展现价值高低而非预算多少来进行展现机会的分配，在这种设置下，往往会产生一个更好的均衡点。

然而，实际上 CoopIL 所采取的这种通过价值进行分配的策略可能牺牲掉部分广告主的利益来实现社会福利最大化，特别是当某些广告主的展现价值一直大于其他广告主时。此外，图5.2e 和图5.2f显示协作也会导致较低的平台收入。例如，当 $C_0 = 1$ 时, CoopIL 得到的平台收入为 28, 26, 30, 这远小于 CompIL 得到的 69, 85, 71。这是因为 CoopIL 所训练的智能体学习到一种共谋策略，通过出低价来预留更多的预算，这会导致平台收入降低。

竞争方式与合作方式所导致智能体的行为趋向于两种极端: 在预算不平衡的情况下，竞争方式导致寡头现象的产生，进而导致较低的社会福利。相比之下，合作的方式获得了更优的社会福利，但这是以减少平台收入为代价的，可能会为了获得更好的社会福利而牺牲部分广告主的利益。

5.4 方法

本文提出一个解决多智能体自动竞价问题的强化学习框架，称为 MAAB。MAAB 主要包括以下两个部分：1) 基于温度调控的奖励分配机制以平衡合作与竞争，2) 多个门槛智能体和门槛门控机制来提升平台收入。

5.4.1 基于温度调控的奖励分配

本节提出一种基于温度调控的奖励分配机制 (Temperature Regularized Credit Assignment, TRCA)，其主要思想是为每个智能体的环境奖励 r^i 设置一个权重参数 α^i。权重参数度量了每个智能体对总体奖励的贡献，因此重新分配给每个智能体的奖励值可以写成

$$r^i_{\text{TRCA}} = \alpha^i \cdot r_{\text{tot}}, \tag{5.4}$$

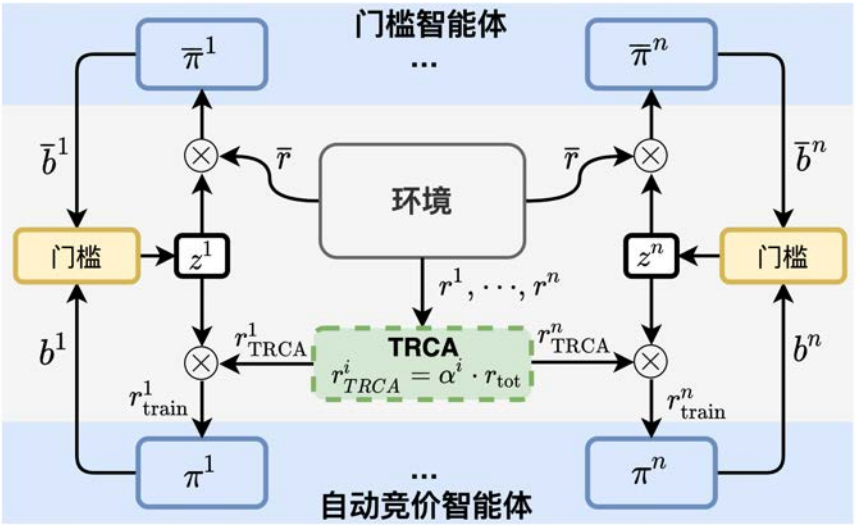


图 5.3　MAAB 架构示意图。

其中 $\alpha^i = \frac{\exp\{b^i/\tau\}}{\sum_j^n \exp\{b^j/\tau\}} \in [0,1]$ 通过 softmax 进行设置，且满足 $\sum_{i=1}^n \alpha^i = 1$。$r_{tot} = \sum_{i=1}^n r^i$ 定义为所有智能体环境奖励之和，衡量了该次展现机会的分配对于整体社会福利的贡献。τ 是平衡竞争和合作的温度参数。

TRCA 背后的直觉是每个广告展现机会的分配结果是通过 r_{tot} 进行衡量的，而自动竞价智能体的出价表达了它们赢得这种展现机会的意愿。智能体中最高的出价主导了 r_{tot} 的大小，而较低的出价则对 r_{tot} 的影响不大。然而，如果没有给出低价的智能体，最高出价可能会因为缺乏竞争而降低。因此，分配给每个智能体的奖励 r^i_{TRCA} 应该随着其出价而增加，本方案中提出的 TRCA 以 softmax 的方式实现 α^i 与 b^i 的正相关性。

此外，式5.4中的参数 τ 使得智能体之间的竞争与合作关系可以同时存在，并且可以很方便地让我们在两者之间实现一个平衡。为了阐述这一点，我们研究 τ 如何影响智能体的行为。为简化分析，我们研究单轮拍卖。假定两个智能体的展现价值满足 $v^1 > v^2$，令 $\mathbf{b}_* = (b^1_*, b^2_*)$ 表示两个智能体的最优出价。那么若两个智能体之间的关系是合作，则智能体 2 应当出更低的价格。形式化地，有以下的定理：

定理 4. *考虑单轮拍卖，竞价环境中仅有两个智能体，其中 $v^1 > v^2$。对于 TRCA 方法，令 $\tau > 0$。若 $v^1 \geq 2v^2$ 或*

$$\tau \geq \frac{\log\left(2v^2/v^1 - 1\right)}{b_{min} - b_{max}}, (v^1 < 2v^2) \tag{5.5}$$

那么 $b^2_* \leq b^1_*$。

证明: 令 $\mathbf{b} = (b^1, b^2)$，对于智能体 2, 其在不同情况下优化的目标为:

$$\begin{cases} \alpha^2(\mathbf{b})v^1 & \text{if } b^1 \geq b^2 \text{ (情况 A)} \\ \alpha^2(\mathbf{b})v^2 & \text{otherwise (情况 B)} \end{cases}$$

定义情况 A 和情况 B 的解集分别为: $\mathbf{B}_A = \{(b_A^1, b_A^2) | b_{min} \leq b_A^2 \leq b_A^1 \leq b_{max}\}, \mathbf{B}_B = \{(b_B^1, b_B^2) | b_{min} \leq b_B^1 \leq b_B^2 \leq b_{max}\}$。目标是找到能使得最优解为情况 A 的条件，因此目标转变为:

$$\max_{\mathbf{b}_A \in \mathbf{B}_A} \alpha^2(\mathbf{b}_A)v^1 \geq \max_{\mathbf{b}_A \in \mathbf{B}_B} \alpha^2(\mathbf{b}_B)v^2.$$

可以写为:

$$\begin{aligned} g(\tau) &= \frac{\max_{\mathbf{b}_A \in \mathbf{B}_A} \alpha^2(\mathbf{b}_A)v^1}{\max_{\mathbf{b}_B \in \mathbf{B}_B} \alpha^2(\mathbf{b}_B)v^2} \\ &= \frac{v^1}{2v^2} \frac{1}{\max_{\mathbf{b}_B \in \mathbf{B}_B} \alpha^2(\mathbf{b}_B)} \\ &= \frac{v^1}{2v^2} \min_{\mathbf{b}_B \in \mathbf{B}_B} \frac{\exp\{b_B^1/\tau\} + \exp\{b_B^2/\tau\}}{2\exp\{b_B^2/\tau\}} \\ &= \frac{v^1}{2v^2} \min_{\mathbf{b}_B \in \mathbf{B}_B} \left(\exp\{(b_B^1 - b_B^2)/\tau\} + 1\right) \\ &= \frac{v^1}{2v^2} \left(\exp\{(b_{min} - b_{max})/\tau\} + 1\right) \\ &\geq 1 \end{aligned} \tag{5.6}$$

当 $v^1 \geq 2v^2$, 总有 $g(\tau) \geq 1$。否则，总可以通过式(5.6)设置 $\tau \geq \frac{\log\left(2v^2/v^1 - 1\right)}{b_{min} - b_{max}}$ 使得情况 A 成立。 □

5.4.2 门槛智能体与门槛门控机制

结合上述 TRCA，本节进一步提出提升平台收入的方法，称为 MAAB。MAAB 的整体架构如图5.3所示。MAAB 引入了多个门槛智能体 $\{\bar{\pi}^i\}_{i=1}^n$，每个门槛智能体 $\bar{\pi}^i$ 旨在为相应的自动竞价智能体 π^i 设置一个个性化的竞价门槛 $\bar{b}^i$。每个门槛智能体 $\bar{\pi}^i$ 与其相应的自动竞价智能体 π^i 共享相同的观测值。在每个时间步，门槛智能体和自动竞价智能体分别给出他们的出价 $\{\bar{b}^i\}_{i=1}^n$ 和 $\{b^i\}_{i=1}^n$，但只有自动竞价智能体的出价 $\{b^i\}_i^n$ 会参与到拍卖中，接着环境返回奖励 $\{r^i\}_{i=1}^n$ 以及相应的广告展现成本，环境奖励经过 TRCA 进行重新分配后，得到 $\{r_{\text{TRCA}}^i\}_{i=1}^n$。成本被视为 n 个门槛智能体共享的奖励 $\bar{r}$。为了防止竞价门槛较大，本方案提出了一种奖励机制，称为 “门槛门控机制”。门槛门控机制为每一对自动竞价智能体和门槛智能体输出标量 $z^i = z(\bar{b}^i b^i) \in \{0, 1\}$，用以指示每个自动竞价智能体的出价是否超过相应门槛智能体设置的竞价门槛。门槛门控机制的基本规则在式5.7中给出。最后, $r_{\text{train}}^i = z^i \cdot r_{\text{TRCA}}^i$ 和 $\bar{r}_{train}^i = z^i \cdot \bar{r}$ 分别用于优化 π^i 和 $\bar{\pi}^i$。注意，门槛智能体仅在训练期间引入，在执行期间完全移除。

$$z(b^i, \bar{b}^i) = \begin{cases} 1 & \text{if } b^i \geq \bar{b}^i \\ 0 & \text{otherwise .} \end{cases} \tag{5.7}$$

门槛智能体和自动竞价智能体通过对抗的方式同时训练。π^i 的训练过程是优化 $\bar{b}^i$ 以最大化收入，而 π^i 则由于合作关系希望降低出价以保留更多的预算。门槛门控机制通过强制门槛智能体的出价成为自动竞价智能体出价的最大下界来为这两个不同的目标建立联系。一方面，如果门槛智能体设置的竞价门槛 $\bar{b}^i$ 远高于对应自动竞价智能体的出价，即 $\bar{b}^i > b^i$, 那么基于门槛门控机制的规则，两者都无法获得奖励。因此，门槛智能体将学会降低出价门槛，而自动出价智能体将相应提高其出价，直到双方就出价达成一致，即 $b^i = \bar{b}^i$, 在这种情况下，双方都可以获得奖励。另一方面，如果在初始阶段 $\bar{b}^i$，双方都可以获得奖励。然而下一阶段门槛智能体会探索是否可以提高 $\bar{b}^i$ 以增加平台的收入，而 b^i 将由于 TRCA 的合作关系而降低。当 $\bar{b}^i$ 增加到与 b^i 降低到的同一水平时，两者出现均衡。总之，本方案提出的门槛智能体和门槛门控机制可将自动竞价智能体的出价提高到适当的水平来提高平台的收入。

5.5 实验

我们在两个智能体的竞价环境中对所提方法进行实验。该部分首先介绍实验环境以及设置，然后给出相应的实验结果，并在最后给出对于门槛智能体的消融实验分析。

5.5.1 实验设置

环境设置. 在该任务中, 两个自动竞价智能体基于各自的预算约束对按序而来的广告展现机会进行竞价。在初始阶段，两个智能体以一定量的预算进行初始化。剩余预算，采样的展现价值，以及剩余的竞价机会编码为智能体的观测 $o^i = (B^i, v^i, ts^i)$，每个智能体基于观测进行出价 b^i。出价较高的智能体将赢得竞标，并将采样的展现价值作为自身奖励，而另一智能体的出价将被视为该次展现的成本。最终，两个智能体更新各自的预算以及竞价机会，并开始下一轮的竞价。拍卖序列的长度设置为 $T = 100$。展现价值值是从均值为 0.5，方差为 1 的正态分布中采样的。为防止自动出价智能体给出较低的出价，本文将保留价设置为 0.1。

评估指标. 本文使用一下两种评估指标: 1) 社会福利, 这是通过将两个广告主的累积价值求和而得到；2) 平台收入，这是依据 GSP 机制计算的广告展现累积代价。

预算约束. 两个智能体的初始预算分别被设置为 $B^1 = T \times C_0 \times ratio$ 以及 $B^2 = T \times C_0 \times (1 - ratio)$。$C_0$ 为预算约束参数，$ratio$ 控制两个智能体预算的均衡。本文分别使用 $C_0 = 1/8, 1/4, 1/2, 1$ 以及 $ratio = 0.3, 0.5, 0.7$ 进行实验。

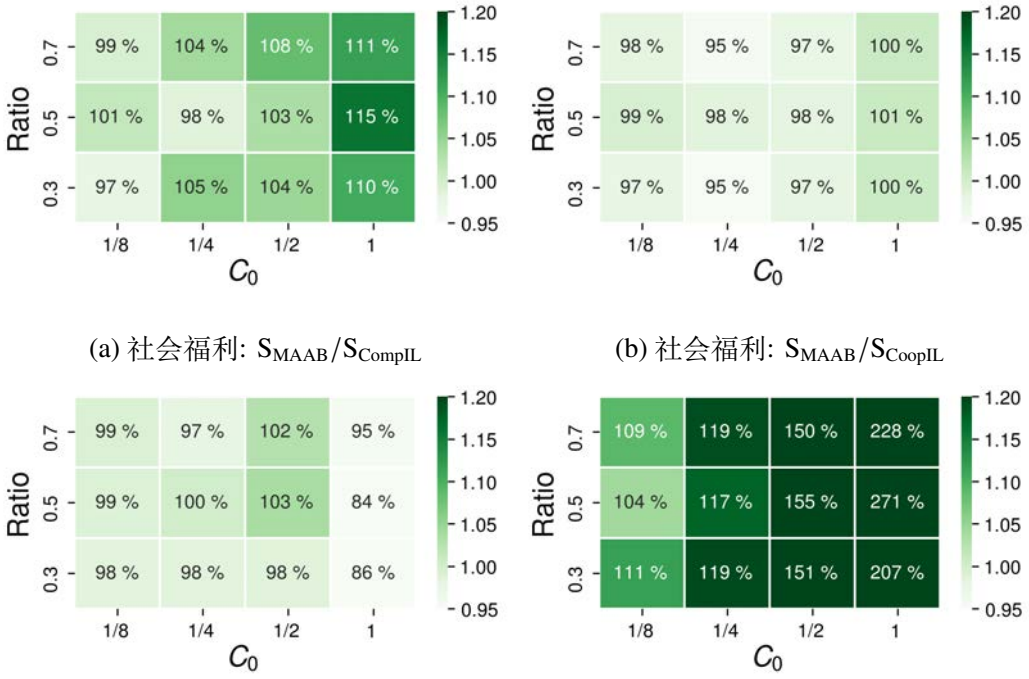
实现细节. 每个智能体的 Q 或 $\bar{Q}$ 网络为一个具有 3 层隐藏层的全连接网络，每层网络有 64 个节点。在训练阶段，每个智能体通过 ϵ-greedy 策略进行探索，ϵ 在 50k 个时间步中线性地从 1.0

退火到 0.05。本文从经验缓冲池中均匀地采样 32 条轨迹，每条轨迹包含最近的 5000 条轨迹，并且在每一条轨迹采样结束时进行一次 Q 网络的训练。为了 $\bar{Q}$ 网络更快地收敛，$\bar{Q}$ 在每条轨迹采样结束后更新两次。Q 和 $\bar{Q}$ 的目标网络在每 200 条训练轨迹后更新一次。

为了加速训练，Q 和 $\bar{Q}$ 分别由所有自动竞价智能体和门槛智能体共享。因此每个智能体的 one-hot 编码也作为每个智能体观测值的输入。所有的神经网络均通过 RMSprop 进行训练，学习率为 0.0005。本文设置 $\gamma = 0.99$ 以及 $\tau = 4$。区间 $[0, 5]$ 被离散化为 21 等距的值，每个智能体 21 个离散化的区间内选择动作。

5.5.2 实验结果

本文将 MAAB 与 CompIL 和 CoopIL 进行了比较。图5.4 显示了 3 个方法在 12 种不同的参数设置下的社会福利以及平台收入比较。每一个有颜色的网格表示相对性能 S_1/S_2 (或 P_1/P_2)，代表就社会福利 (或平台收入) 而言，方法 1 的性能除以方法 2 相应的性能。$S_1/S_2 > 100\%$ (或 $P_1/P_2 > 100\%$) 表示方法 1 就社会福利 (或平台的收入) 而言优于方法 2。



(a) 社会福利: S_{MAAB}/S_{CompIL}　　(b) 社会福利: S_{MAAB}/S_{CoopIL}

(c) 平台收入: P_{MAAB}/P_{CompIL}　　(d) 平台收入: P_{MAAB}/P_{CoopIL}

图 5.4　在不同的 C_0 和 *ratio* 参数设置下，MAAB 与 CompIL 以及 CoopIL 的性能比较。

与 CompIL 相比 (图5.4a 以及图5.4c)，就社会福利而言，MAAB 在绝大多数设置下优于 CompIL，这是由于 TRCA 方法所引入的协作机制；并且在大多数设置下，MAAB 就平台收入而言其性能与 CompIL 比肩。例如，当 $C_0 = 1/2$ 时，S_{MAAB}/S_{CompIL}=108%, 103%, 104% 且

P_{MAAB}/P_{CompIL}=102%, 103%, 98%。然而，当 $C_0 = 1$ (P_{MAAB}/P_{CompIL}=95%, 84%, 86%) 时，与 CompIL 相比，MAAB 达到的平台收入更低。这是因为 $C_0 = 1$ 是一种预算无约束的设置，在这种情形下，双方智能体均无法花光其预算，因此智能体将会贪婪地出高价，造成在每次拍卖中需花费极高的成本。

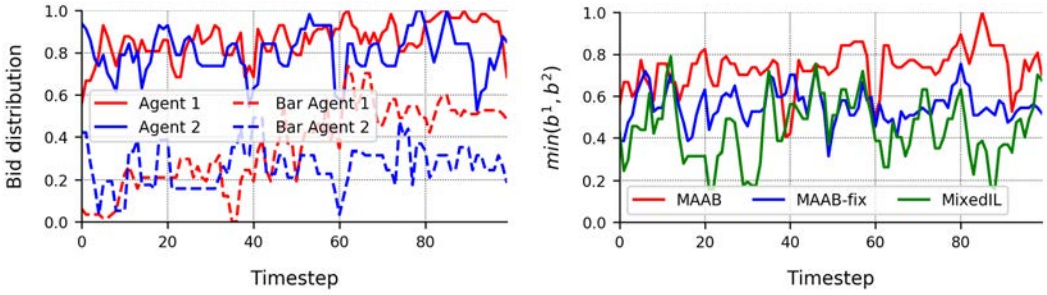
在图5.4b 和图5.4d中, 本文比较了 MAAB 和完全协作的 CoopIL 方法。可以发现，就平台收入而言, MAAB 在所有设置下均远远高于 CoopIL。特别是在充足预算的情况下 ($C_0 = 1$), MAAB 所得到的平台收入是 CoopIL 的两倍多 (P_{MAAB}/P_{CoopIL}=228%, 271%, 207%), 这充分表明了本文所提出的门槛智能体的有效性。更重要的是，如图5.4b所示, 与 CoopIL 相比，由于门槛门控机制的使用，门槛智能体的引入在没有降低社会福利的情况下提高了平台收入。(例如, 当 C_0=1, S_{MAAB}/S_{CoopIL}=100%, 101%, 100%)

5.5.3 消融分析

本部分探究门槛智能体在提高平台收入上的有效性。为了达到该目的，本部分将 MAAB 与以下两种方法进行比较: 1) MixedIL: 该方法通过在 MAAB 中移除门槛智能体从而得到; 2) MAAB-fix: 通过在 MAAB 中固定门槛智能体的动作为一个启发式的固定值得到，本部分将启发值设置为 $\bar{b} = 1$。所有方法均训练 2 百万个时间步。

为了说明门槛智能体在提高平台收入上的有效性，本文选了一条轨迹并绘出两个智能体在不同时刻的出价。实验中使用 $C_0 = 1$ 以及 $ratio = 0.5$。如图5.5a所示, 通过比较相同颜色的实线与虚线，可以发现门槛智能体的出价严格低于其对应的自动竞价智能体。这表明门槛门控机制的有效性，因为在式5.7中，若 $b^i \leq \bar{b}^i$，则双方智能体均无法获得环境奖励。有趣的是，本文的结果表明门槛智能体所设置的竞价门槛与其对应的自动竞价智能体的出价具有类似的模式。如图5.5b所示，智能体 1 (蓝线实线) 与门槛智能体 1 (蓝色虚线) 均在时间步 19 处有一个出价高峰，并且在时间步 60 处有一个出价低谷。对于智能体 2（红色实线）以及其对应的门槛智能体 2，类似的模式在时间步 40 和 62 处可以看到。这些模式表明门槛智能体与门槛门控机制可以基于智能体的展现价值以及预算等信息自适应地调节竞价门槛，与 MAAB-fix 这种没有使用额外信息的固定竞价门槛方法相比，这种能够自适应地调节竞价门槛的方法呈现出了非常大的优势。

在图5.5b中, 本文绘出了 MAAB, MAAB-fix ($\bar{b} = 1$) 以及 MixedIL 在不同时间步的成本。可以发现具有竞价门槛 (MAAB, MAAB-fix) 的方法在成本上优于没有竞价门槛的方法 (MixedIL), 这表明 $\bar{b}^i$ 可以提高平台的收入。此外，MAAB 的成本高于 MAAB-fix，这表明自适应地设置动态门槛值要优于静态值。



(a) MAAB 中不同智能体的出价　　(b) 三个方法在不同时刻的成本

图 5.5　MAAB 的出价以及在选定的一条轨迹中三种方法在不同时间步的成本。所有的方法都经过 2 百万个时间步的训练。

5.6　本章小结

自动竞价已经成为优化广告主的广告展现目标的重要工具。本部分为自动竞价问题提出了一种多智能体强化学习框架，称为 MAAB。MAAB 主要包括以下两点贡献: 1) 提出温度调控的奖励分配机制，从而建立了自动竞价智能体之间的混合协作竞争关系，2) 引入了门槛智能体以及一种门槛门控机制，通过一种对抗的方式进行训练，从而提高了平台的收入。本文在两智能体竞价环境中的实验表明，MAAB 在可以实现较高的社会福利以及保证平台收入。未来的工作包括根据实时信息动态调整温度参数来扩展 TRCA。此外，为了加快门槛智能体的策略收敛速度，门槛门控机制下的奖励分配规则仍需要进一步研究。

第六章　总结与展望

6.1　工作总结

深度强化学习最近在许多方面都取得了成功[30,36]，深度强化学习继续发展的下一步自然是在多智能体场景下发挥其能力。然而，在多智能体环境中学习本质上是非常有难度的。特别地，在多智能体环境中的主要挑战包括联合动作空间的大小随智能体的数量成指数增长[40,41]，由智能体策略的改变导致的不稳定环境[47,74]以及合作场景中的多智能体全局奖励分配问题[40,41]。这些挑战使得将所有智能体视为单个元智能体的完全集中式方法，以及通过将其他智能体视为环境的一部分来分别训练每个智能体的完全分散式方法都变得非常困难[39–41,45]。

针对这些挑战，本文首先回顾了解决上述问题的已有工作。针对多智能体联合动作空间随智能体数量指数增长的问题，已有工作通常采用“集中训练、分散执行”的范式。而对于环境的非静态性问题，COMA、QMIX 以及 QTRAN 通过在训练阶段引入集中价值函数，并引入全局信息来减轻环境非静态性所导致的问题。而对于协作场景下的奖励分配问题，COMA 基于差分奖励的思路，通过计算反事实的基准解决奖励分配问题，VDN 以及 QMIX 则是一种基于值分解方法的隐式奖励分配方案。

上述已有工作均为基于 CTDE 框架下的多智能体强化学习算法，一般来说，基于 CTDE 框架的多智能体方法一般需要估计一个集中价值函数，该集中价值函数估计的准确性会大大影响到算法的表现。为解决值函数估计困难的问题，本文提出了一个新的具有较高样本效率，基于 CTDE 框架的多智能体强化学习方法，称为 SMIX(λ)。SMIX(λ) 通过基于异策略的集中价值函数学习方法改进了集中价值函数估计过程，该方法消除了在训练过程中显式依赖集中贪婪行为 (CGB) 假设的需要，并且引入的 λ-回报[1] 可以更好地平衡集中价值函数估计中的偏差和方差。SMIX(λ) 使用异策略学习机制是由重要性采样驱动的，但是通过经验回放机制来实现，这种机制与先前的单智能体学习中的 Q(λ) 方法有着密切的联系。通过结合这些要素，SMIX(λ) 方法有效地提高了样本效率并稳定了训练过程。以星际争霸多智能体挑战赛 (SMAC)[58] 为基准，SMIX(λ) 可以在多数场景中均获得最好的性能。此外，通过将现有的 CTDE 型多智能体强化学习算法的集中价值函数估计器替换为本文新提出的 SMIX(λ)，可以发现现有 CTDE 型多智能体强化学习算法可以获得显著的性能提升。

本文的第二个主要部分考虑多智能体强化学习在现实问题中的应用问题。本文以广告自动竞价问题为例[59]，考虑从多智能体系统的角度来对该问题进行建模。为此，我们针对自动竞价问题提出了一种混合协作竞争的多智能体强化学习框架，称为 MAAB。在该框架中，本文首先提出了一种基于温度调控的奖励分配方法，实现了自动竞价智能体之间协作与竞争关系的折中，

从而建立了自动竞价智能体之间混合协作竞争的关系。其次，受最优拍卖中[60–62] 的保留价启发，本文设计了门槛智能体来为每一个自动竞价智能体学习一个具有个性化的竞价门槛，以减少平台收入的降低。本文在两个智能体的自动竞价环境中对本文所提方法进行验证，实验表明本文方法可以在提高社会福利的同时，减少平台收入的降低。

6.2 未来展望

上述工作只是本文对多智能体强化学习协作问题的初步探讨。然而，由于多智能体协作问题受到非常多其他因素的影响，仍有许多不完善的地方有待进一步讨论。这里将进一步研究的工作归纳为如下几个方面：

第一，异质智能体系统协作效率低。许多多智能体系统中都存在异质智能体，即系统内存在个体类型不同的智能体。例如在军事作战中，作战兵种往往具有多样性并且功能差异较大 (例如步兵和坦克)，个体间的差异往往会提高协同训练的难度。这种情况下，往往需要按照个体的特性和功能学习具有差异化且互补的策略。在未来的工作中，可以从个体特征的先验信息出发，考虑为不同智能体学习具有差异化的策略参数。

第二，大规模智能体难以集中训练的问题。目前多智能体强化学习领域广泛采用“集中训练，分散执行”[41,50] 的范式，即需训练集中价值函数以缓解环境的非静态性[91]，在学习该网络时，一般需要将所有智能体的观测信息 $(o^1, \cdots, o^n)$ 作为输入，然而该信息随着智能体数量的增加呈现线性增张，造成集中价值函数假设空间的复杂性。因此可以考虑从集中式价值函数 $Q((o^1, \cdots, o^n), \mathbf{a})$ 的假设空间出发，通过为该函数设计适当的结构，使之满足“组合不变性”[92]。这一动机来源于在学习该函数时，不同智能体观测值的输入次序不应当影响到 $Q((o^1, \cdots, o^n), \mathbf{a})$ 的值。这一合理假设往往可以大大简化神经网络的结构，让神经网络得以从“学习次序”中解放出来，进而提高学习效率。

第三，多智能体强化学习难以落地的问题。强化学习一般需要精确的模拟器来与之交互产生数据，进而通过交互数据来训练以最大化累积奖励。但在现实应用中研制精确的模拟器往往需要投入大量的人力和物力。此外，真实的多智能体环境往往是随时间动态变化的，这对强化学习的落地问题提出了更高的挑战。为解决真实环境与模拟环境差异的问题，可以考虑采用数据驱动的离线强化学习方法，通过已有经验数据进行学习，而无需与环境进行交互。

参考文献

[1] Sutton R S, Barto A G. Reinforcement learning: An introduction[M]. MIT press, 2018.

[2] Caruana R, Niculescu-Mizil A. An empirical comparison of supervised learning algorithms[C]. Proceedings of Proceedings of the 23rd international conference on Machine learning, 2006. 161–168.

[3] Barlow H B. Unsupervised learning[J]. Neural computation, 1989, 1(3):295–311.

[4] Bishop C, Nasrabadi N M. Pattern Recognition and Machine Learning[J]. Journal of Electronic Imaging, 2007, 16.

[5] Goodfellow I, Bengio Y, Courville A, et al. Deep learning[M], volume 1. MIT press Cambridge, 2016.

[6] Voulodimos A, Doulamis N, Doulamis A, et al. Deep learning for computer vision: A brief review[J]. Computational intelligence and neuroscience, 2018, 2018.

[7] Tan M, Le Q V. Efficientnet: Rethinking model scaling for convolutional neural networks[J]. arXiv preprint arXiv:1905.11946, 2019..

[8] Ren S, He K, Girshick R, et al. Faster r-cnn: Towards real-time object detection with region proposal networks[J]. IEEE transactions on pattern analysis and machine intelligence, 2016, 39(6):1137–1149.

[9] He K, Zhang X, Ren S, et al. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification[C]. Proceedings of Proceedings of the IEEE international conference on computer vision, 2015. 1026–1034.

[10] Chung J, Gulcehre C, Cho K, et al. Empirical evaluation of gated recurrent neural networks on sequence modeling[J]. arXiv preprint arXiv:1412.3555, 2014..

[11] Povey D, Ghoshal A, Boulianne G, et al. The Kaldi speech recognition toolkit[C]. Proceedings of IEEE 2011 workshop on automatic speech recognition and understanding. IEEE Signal Processing Society, 2011.

[12] Park D S, Chan W, Zhang Y, et al. Specaugment: A simple data augmentation method for automatic speech recognition[J]. arXiv preprint arXiv:1904.08779, 2019..

[13] Chowdhary K. Natural language processing[M]. . Proceedings of Fundamentals of Artificial Intelligence. Springer, 2020: 603–649.

[14] Deng L, Liu Y. Deep learning in natural language processing[M]. Springer, 2018.

[15] Young T, Hazarika D, Poria S, et al. Recent trends in deep learning based natural language processing[J]. ieee Computational intelligenCe magazine, 2018, 13(3):55–75.

[16] Li H. Deep learning for natural language processing: advantages and challenges[J]. National Science Review, 2017..

[17] Gullapalli V, Franklin J A, Benbrahim H. Acquiring robot skills via reinforcement learning[J]. IEEE Control Systems Magazine, 1994, 14(1):13–24.

[18] Riedmiller M, Gabel T, Hafner R, et al. Reinforcement learning for robot soccer[J]. Autonomous Robots, 2009, 27(1):55–73.

[19] Ueda K, Hatono I, Fujii N, et al. Reinforcement learning approaches to biological manufacturing systems[J]. CIRP annals, 2000, 49(1):343–346.

[20] Mahadevan S, Theocharous G. Optimizing Production Manufacturing Using Reinforcement Learning.[C]. Proceedings of FLAIRS Conference, volume 372, 1998. 377.

[21] Scheffler K, Young S. Automatic learning of dialogue strategy using dialogue simulation and reinforcement learning[C]. Proceedings of Proceedings of the second international conference on Human Language Technology Research. Citeseer, 2002. 12–19.

[22] Shi J C, Yu Y, Da Q, et al. Virtual-taobao: Virtualizing real-world online retail environment for reinforcement learning[C]. Proceedings of Proceedings of the AAAI Conference on Artificial Intelligence, volume 33, 2019. 4902–4909.

[23] Zhang W, Dietterich T G. A reinforcement learning approach to job-shop scheduling[C]. Proceedings of IJCAI, volume 95. Citeseer, 1995. 1114–1120.

[24] Chinchali S, Hu P, Chu T, et al. Cellular Network Traffic Scheduling With Deep Reinforcement Learning.[C]. Proceedings of AAAI, 2018. 766–774.

[25] Wu D, Chen X, Yang X, et al. Budget constrained bidding by model-free reinforcement learning in display advertising[C]. Proceedings of Proceedings of the 27th ACM International Conference on Information and Knowledge Management, 2018. 1443–1451.

[26] Cai H, Ren K, Zhang W, et al. Real-time bidding by reinforcement learning in display advertising[C]. Proceedings of Proceedings of the Tenth ACM International Conference on Web Search and Data Mining, 2017. 661–670.

[27] Zhang W, Yuan S, Wang J. Optimal real-time bidding for display advertising[C]. Proceedings of Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining, 2014. 1077–1086.

[28] Busoniu L, Babuska R, De Schutter B. A comprehensive survey of multiagent reinforcement learning[J]. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), 2008, 38(2):156–172.

[29] Silver D, Schrittwieser J, Simonyan K, et al. Mastering the game of go without human knowledge[J]. Nature, 2017, 550(7676):354.

[30] Vinyals O, Babuschkin I, Czarnecki W M, et al. Grandmaster level in StarCraft II using multi-agent reinforcement learning[J]. Nature, 2019, 575(7782):350–354.

[31] Berner C, Brockman G, Chan B, et al. Dota 2 with large scale deep reinforcement learning[J]. arXiv preprint arXiv:1912.06680, 2019..

[32] Cui J, Liu Y, Nallanathan A. Multi-Agent Reinforcement Learning-Based Resource Allocation for UAV Networks[J]. IEEE Transactions on Wireless Communications, 2019, 19(2):729–743.

[33] Perolat J, Leibo J Z, Zambaldi V, et al. A multi-agent reinforcement learning model of common-pool resource appropriation[J]. Advances in Neural Information Processing Systems, 2017, 30:3643–3652.

[34] Wu J, Xu X, Zhang P, et al. A novel multi-agent reinforcement learning approach for job scheduling in Grid computing[J]. Future Generation Computer Systems, 2011, 27(5):430–439.

[35] Jin J, Song C, Li H, et al. Real-time bidding with multi-agent reinforcement learning in display advertising[C]. Proceedings of Proceedings of the 27th ACM International Conference on Information and Knowledge Management, 2018. 2193–2201.

[36] Mnih V, Kavukcuoglu K, Silver D, et al. Human-level control through deep reinforcement learning[J]. Nature, 2015, 518(7540):529.

[37] Schulman J, Wolski F, Dhariwal P, et al. Proximal policy optimization algorithms[J]. arXiv preprint arXiv:1707.06347, 2017..

[38] AlQuraishi M. AlphaFold at CASP13[J]. Bioinformatics, 2019, 35(22):4862–4865.

[39] Foerster J, Nardelli N, Farquhar G, et al. Stabilising experience replay for deep multi-agent reinforcement learning[C]. Proceedings of International Conference on Machine Learning, volume 70. JMLR. org, 2017. 1146–1155.

[40] Foerster J N, Farquhar G, Afouras T, et al. Counterfactual multi-agent policy gradients[C]. Proceedings of Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, 2018.

[41] Rashid T, Samvelyan M, Witt C S, et al. QMIX: Monotonic Value Function Factorisation for Deep Multi-Agent Reinforcement Learning[C]. Proceedings of International Conference on Machine Learning, 2018. 4292–4301.

[42] Hernandez-Leal P, Kartal B, Taylor M E. Is multiagent deep reinforcement learning the answer or the question? A brief survey[J]. arXiv preprint arXiv:1810.05587, 2018..

[43] Wei E, Wicke D, Freelan D, et al. Multiagent soft q-learning[J]. arXiv preprint arXiv:1804.09817, 2018..

[44] Da Silva B C, Basso E W, Bazzan A L, et al. Dealing with non-stationary environments using context detection[C]. Proceedings of Proceedings of the 23rd international conference on Machine learning, 2006. 217–224.

[45] Tan M. Multi-agent reinforcement learning: Independent vs. cooperative agents[C]. Proceedings of International Conference on Machine Learning, 1993. 330–337.

[46] Oliehoek F A, Amato. A concise introduction to decentralized POMDPs[M], volume 1. Springer, 2016.

[47] Lowe R, Wu Y, Tamar A, et al. Multi-agent actor-critic for mixed cooperative-competitive environments[C]. Proceedings of Advances in Neural Information Processing Systems, 2017. 6379–6390.

[48] Sunehag P, Lever G, Gruslys A, et al. Value-decomposition networks for cooperative multi-agent learning based on team reward[C]. Proceedings of International Conference on Autonomous Agents and MultiAgent Systems, 2018. 2085–2087.

[49] Son K, Kim D, Kang W J, et al. QTRAN: Learning to Factorize with Transformation for Cooperative Multi-Agent Reinforcement Learning[C]. Proceedings of International Conference on Machine Learning, 2019. 5887–5896.

[50] Foerster J, Farquhar G, Afouras T, et al. Counterfactual multi-agent policy gradients[J]. arXiv preprint arXiv:1705.08926, 2017..

[51] Wang Z, Bapst V, Heess N, et al. Sample efficient actor-critic with experience replay[J]. arXiv preprint arXiv:1611.01224, 2016..

[52] Chang Y H, Ho T, Kaelbling L P. All learning is local: Multi-agent learning in global reward games[C]. Proceedings of Advances in Neural Information Processing Systems, 2004. 807–814.

[53] Tumer K, Agogino A. Distributed agent-based air traffic flow management[C]. Proceedings of Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems, 2007. 1–8.

[54] Lillicrap T P, Hunt J J, Pritzel A, et al. Continuous control with deep reinforcement learning[J]. arXiv preprint arXiv:1509.02971, 2015..

[55] Bellman. Dynamic Programming[M]. Rand Corporation research study, Princeton University Press, 1957.

[56] Tokdar S T, Kass R E. Importance sampling: a review[J]. Wiley Interdisciplinary Reviews: Computational Statistics, 2010, 2(1):54–60.

[57] Harutyunyan A, Bellemare M G, Stepleton T, et al. Q (λ) with Off-Policy Corrections[C]. Proceedings of Proceedings of the International Conference on Algorithmic Learning Theory. Springer, 2016. 305–320.

[58] Samvelyan M, Rashid T, Witt C, et al. The starcraft multi-agent challenge[C]. Proceedings of International Conference on Autonomous Agents and MultiAgent Systems, 2019. 2186–2188.

[59] Aggarwal G, Badanidiyuru A, Mehta A. Autobidding with constraints[C]. Proceedings of WINE. Springer, 2019. 17–30.

[60] Mohri M, Medina A M. Learning theory and algorithms for revenue optimization in second price auctions with reserve[C]. Proceedings of ICML. PMLR, 2014. 262–270.

[61] Yuan S, Wang J, Chen B, et al. An empirical study of reserve price optimisation in real-time bidding[C]. Proceedings of SIGKDD, 2014. 1897–1906.

[62] Ostrovsky M, Schwarz M. Reserve prices in internet advertising auctions: A field experiment[C]. Proceedings of EC, 2011. 59–60.

[63] Konda V R, Tsitsiklis J N. Actor-critic algorithms[C]. Proceedings of Advances in neural information processing systems, 2000. 1008–1014.

[64] Watkins C J C H. Learning from delayed rewards[J]. 1989..

[65] Schulman J, Levine S, Abbeel P, et al. Trust region policy optimization[C]. Proceedings of International Conference on Machine Learning, 2015. 1889–1897.

[66] Schulman J, Moritz P, Levine S, et al. High-dimensional continuous control using generalized advantage estimation[J]. arXiv preprint arXiv:1506.02438, 2015..

[67] Rummery G A, Niranjan M. On-line Q-learning using connectionist systems[M], volume 37. University of Cambridge, Department of Engineering Cambridge, England, 1994.

[68] Silver D, Lever G, Heess N, et al. Deterministic policy gradient algorithms[C]. . 2014.

[69] Tampuu A, Matiisen T, Kodelja D, et al. Multiagent cooperation and competition with deep reinforcement learning[J]. PloS one, 2017, 12(4):e0172395.

[70] Leibo J Z, Zambaldi V, Lanctot M, et al. Multi-agent reinforcement learning in sequential social dilemmas[J]. arXiv preprint arXiv:1702.03037, 2017..

[71] Ye D, Zhang M, Yang Y. A multi-agent framework for packet routing in wireless sensor networks[J]. sensors, 2015, 15(5):10026–10047.

[72] Pol E, Oliehoek F A. Coordinated deep reinforcement learners for traffic light control[J]. Proceedings of the Learning, Inference and Control of Multi-Agent Systems, 2016..

[73] Jaderberg M, Czarnecki W M, Dunning I, et al. Human-level performance in 3D multiplayer games with population-based reinforcement learning[J]. Science, 2019, 364(6443):859–865.

[74] Laurent G J, Matignon L, Fort-Piat L, et al. The world of independent learners is not Markovian[J]. International Journal of Knowledge-based and Intelligent Engineering Systems, 2011, 15(1):55–64.

[75] Oliehoek F A, Spaan M T, Vlassis N. Optimal and approximate Q-value functions for decentralized POMDPs[J]. Journal of Artificial Intelligence Research, 2008, 32:289–353.

[76] Kraemer L, Banerjee B. Multi-agent reinforcement learning as a rehearsal for decentralized planning[J]. Neurocomputing, 2016, 190:82–94.

[77] Shalev-Shwartz S, Ben-David S. Understanding machine learning: From theory to algorithms[M]. Cambridge university press, 2014.

[78] Russell S J, Norvig P. Artificial intelligence: a modern approach[M]. Malaysia; Pearson Education Limited,, 2016.

[79] Sutton R, Mahmood A R, Precup D, et al. A new Q (λ) with interim forward view and Monte Carlo equivalence[C]. Proceedings of International Conference on Machine Learning, 2014. 568–576.

[80] Munos R, Stepleton T, Harutyunyan A, et al. Safe and efficient off-policy reinforcement learning[C]. Proceedings of Advances in Neural Information Processing Systems, 2016. 1054–1062.

[81] Liu Q, Li L, Tang Z, et al. Breaking the curse of horizon: Infinite-horizon off-policy estimation[C]. Proceedings of Advances in Neural Information Processing Systems, 2018. 5356–5366.

[82] Fujimoto S, Meger D, Precup D. Off-Policy Deep Reinforcement Learning without Exploration[C]. Proceedings of International Conference on Machine Learning, 2019. 2052–2062.

[83] Hausknecht M, Stone P. Deep recurrent q-learning for partially observable mdps[C]. Proceedings of 2015 AAAI Fall Symposium Series, 2015.

[84] Chung J, Gulcehre C, Cho K, et al. Empirical evaluation of gated recurrent neural networks on sequence modeling[C]. Proceedings of Advances in Neural Information Processing Systems, 2014.

[85] Wang J, Zhang W, Yuan S. Display advertising with real-time bidding (RTB) and behavioural targeting[J]. arXiv preprint arXiv:1610.03013, 2016..

[86] Yuan S, Wang J, Zhao X. Real-time bidding for online advertising: measurement and analysis[C]. Proceedings of International Workshop on Data Mining for Online Advertising, 2013. 1–8.

[87] Marshall R C, Marx L M. Bidder collusion[J]. Journal of Economic Theory, 2007, 133(1):374–402.

[88] Myerson R B. Optimal auction design[J]. Mathematics of operations research, 1981, 6(1):58–73.

[89] Littman M L. Markov games as a framework for multi-agent reinforcement learning[M]. . Proceedings of Machine learning proceedings 1994. Elsevier, 1994: 157–163.

[90] Foerster J, Chen R Y, Al-Shedivat M, et al. Learning with Opponent-Learning Awareness[C]. Proceedings of AAMAS, 2018. 122–130.

[91] Papoudakis G, Christianos F, Rahman A, et al. Dealing with non-stationarity in multi-agent deep reinforcement learning[J]. arXiv preprint arXiv:1906.04737, 2019..

[92] Liu I J, Yeh R A, Schwing A G. PIC: permutation invariant critic for multi-agent deep reinforcement learning[C]. Proceedings of Conference on Robot Learning. PMLR, 2020. 590–602.

致　　谢

两年半的研究生生活犹如白驹过隙，回首这两年的生活，从本科毕业时的对科研一无所知的菜鸟，到现在可以独立完成科研任务的研究生，这样一种身份上的转变不仅意味着学历上的升级，更意味着能力与意志力的提高。然而这一切的成长都离不开对我谆谆教诲的老师，陪伴我科研生活的同门，共同努力的同事，以及对我无私无畏的亲人。

首先，我要感谢我的导师谭晓阳教授。谭老师治学严谨，对数学和理论有着独到的追求，并且一直站在科研的一线。还记得谭老师帮我一字一句地修改论文，从文章的立意到结构，从语法到标点，谭老师这种一丝不苟，严谨负责的精神深深感染着我。回首这两年多的时光，谭老师不仅传授于我前沿的科学理论，还培养了我的科研品味，改变了我看问题的方式。谭老师是我科研的引路人，这一切我会铭记在心，在此向谭老师表达真挚的感谢!

感谢 PARNEC 组里的陈松灿教授、张道强教授、刘学军教授以及黄圣君教授。陈老师不仅学识渊博，为人师表，而且待人热情，每次从办公室路过都可以看到陈老师在为其学生修改论文，这样的学者风范一直深深感染着我。张道强教授、刘学军教授以及黄圣君教授是组里优秀的青年教授，他们优秀的科研水平也一直鼓舞着我做出优秀的科研成果。很庆幸可以融入这样一个年轻且充满浓厚科研氛围的团体，在此感谢组里的各位老师们!

其次，还要感谢我同门的师兄师姐师弟师妹们，特别感谢王宇辉师兄和姚兴虎，与他们一起合作论文的经历令我受益匪浅，至今难忘小明师兄帮我修改论文以及教我做实验设计的经历；姚兴虎是我的室友兼同门，与他点滴相处的时光以及在科研上思想碰撞的瞬间也是一段珍贵的回忆。其次还要感谢宋歌、李尧、甘耀中以及张哲师兄，他们扎实的数学功底，优秀的科研能力一直我学习的榜样，希望他们可以在博士生涯做出优秀的科研成果。还要感谢我的同门方钧婷、袁野还有蒋柯、吴卿源、任国伟等师弟们，深夜常常可以看到他们在实验室的身影，这种刻苦钻研、脚踏实地的精神一直鼓舞着我。

此外，感谢在阿里妈妈实习期间对我进行指导的张知临，喻川，刘翔宇，徐淼等师兄师姐，和他们的每次讨论都让我备受启发，很有幸可以在这样一个优秀的团队中实习。还要感谢上海交大的郑臻哲老师对我进行了细致的学术指导，并为我的论文倾注了许多时间。

最后，感谢我的女朋友，感谢她对我一如既往的支持和理解！感谢我的父母，感谢他们在每一通电话中的关心与鼓舞，感谢他们多年以来对我的支持。

在学期间的研究成果及学术论文情况

攻读研究生学位期间的研究成果

1. **Chao Wen**, Xinghu Yao, Yuhui Wang, Xiaoyang Tan. *SMIX(λ): Enhancing Centralized Value Functions for Cooperative Multi-Agent Reinforcement Learning*. AAAI 2020.
2. **Chao Wen**, Miao Xu, Zhilin Zhang, Zhenzhe Zheng, Yuhui Wang, Xiangyu Liu, Dong Xie, Xiaoyang Tan, Chuan Yu, Jian Xu, Fan Wu, Guihai Chen, Xiaoqiang Zhu. *A Cooperative-Competitive Multi-Agent Framework for Auto-bidding in Online Advertising*. KDD 2021 (在审)
3. Yuhui Wang, Hao He, **Chao Wen**, Xiaoyang Tan. *Truly Proximal Policy Optimization*. arXiv:1903.07940

攻读硕士学位期间专利及软件著作权情况

1. 专利: 一种基于 λ-回报的异策略多智能体强化学习协作方法. 谭晓阳，**文超**，姚兴虎 (专利申请号：CN201911373178.X)
2. 软件著作权: 多步异策略学习的协作式多智能算法软件 (登记号: 2020SR0281996)

研究生期间参与的科研项目

1. 南京航空航天大学校研究生开放基金 (No. Kfjj20191608，项目负责人，结题评议：优秀)
2. 国家自然科学基金 (No.61976115, 61672280, 61732006)
3. 装备预研项目 (No. 6140312020413)
4. 南京航空航天大学 AI+ 项目 (No.56XZA18009)