分类号 TP311.1 学号 18020042

ＵＤＣ　 004.8 密级 公 开

工学硕士学位论文

**基于多智能体强化学习的多目标动态规划**

**技术研究及实现**

|  |  |  |
| --- | --- | --- |
| 硕士生姓名 |  | 王雅洁 |
| 学科专业 |  | 计算机科学与技术 |
| 研究方向 |  | 分布计算技术 |
| 指导教师 |  | 史殿习 研究员 |
| 协助指导教师 |  |  |

**国防科技大学研究生院**

**二〇二〇年十月**

**论文书脊**

（此页只是书脊样式，学位论文不需要印刷本页。）

基于多智能体强化学习的多目标动态规划技术研究及实现 国防科技大学研究生院

**Research and Implementation of Multi-Target Dynamic Programming Technology Based on Multi-Agent Reinforcement Learning**

**Candidate：Yajie Wang**

**Supervisor：Prof. Dianxi Shi**

**Associate Supervisor：**

**A dissertation**

**Submitted in partial fulfillment of the requirements**

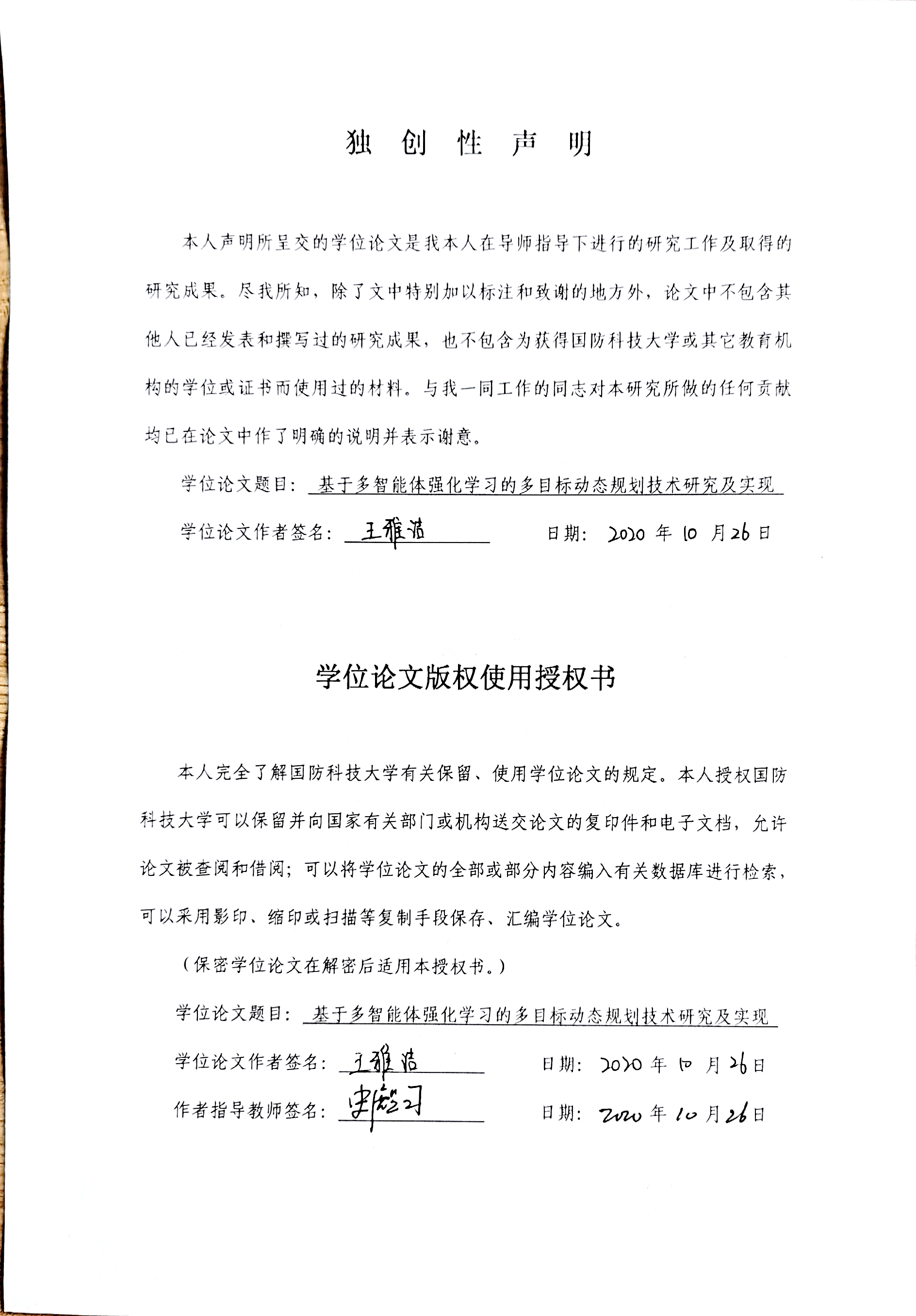
**for the degree of Master of Engineering**

**in Computer science and technology**

**Graduate School of National University of Defense Technology**

**Changsha，Hunan，P.R.China**

**October，2020**



目 录

[摘 要 i](#_Toc56515252)

[ABSTRACT ii](#_Toc56515253)

[1 第一章 绪论 1](#_Toc56515254)

[1.1 研究背景和研究意义 2](#_Toc56515255)

[1.1.1 研究背景 2](#_Toc56515256)

[1.1.2 研究意义 6](#_Toc56515257)

[1.2 国内外研究现状 7](#_Toc56515258)

[1.2.1 智能无人集群 7](#_Toc56515259)

[1.2.2 多目标规划问题 8](#_Toc56515260)

[1.2.3 多智能体深度强化学习 9](#_Toc56515261)

[1.3 研究目标及研究内容 11](#_Toc56515262)

[1.3.1 研究目标 11](#_Toc56515263)

[1.3.2 研究内容 12](#_Toc56515264)

[1.4 论文的组织结构 13](#_Toc56515265)

[1.5 本章小结 14](#_Toc56515266)

[2 第二章 相关工作 15](#_Toc56515267)

[2.1 奖励反馈设计 15](#_Toc56515268)

[2.2 注意力机制 16](#_Toc56515269)

[2.3 多智能体强化学习 17](#_Toc56515270)

[2.3.1 马尔科夫博弈 17](#_Toc56515271)

[2.3.2 深度强化学习 18](#_Toc56515272)

[2.3.3 多智能体强化学习 20](#_Toc56515273)

[2.4 其他相关技术 21](#_Toc56515274)

[2.4.1 Gym强化学习框架 21](#_Toc56515275)

[2.4.2 多智能体粒子环境 21](#_Toc56515276)

[2.5 本章小结 22](#_Toc56515277)

[3 第三章 基于多智能体Actor-Critic框架的多目标规划模型 23](#_Toc56515278)

[3.1 MACTP模型概述 23](#_Toc56515279)

[3.2 多智能体任务建模 23](#_Toc56515280)

[3.3 基于人工势场法的奖励反馈设计 26](#_Toc56515281)

[3.3.1 引力反馈设计 27](#_Toc56515282)

[3.3.2 斥力反馈设计 28](#_Toc56515283)

[3.4 MACTP模型 29](#_Toc56515284)

[3.5 本章小结 31](#_Toc56515285)

[4 第四章 基于分层注意力机制的多智能体强化学习算法 32](#_Toc56515286)

[4.1 AHAC算法框架 32](#_Toc56515287)

[4.2 基于RNN的特征提取机制 33](#_Toc56515288)

[4.3 基于HAM的信息压缩机制 34](#_Toc56515289)

[4.3.1 HAM具体组成 34](#_Toc56515290)

[4.3.2 HAM计算方法 37](#_Toc56515291)

[4.4 多智能体Actor-Critic框架策略更新机制 39](#_Toc56515292)

[4.4.1 多智能体Actor-Critic框架组成 39](#_Toc56515293)

[4.4.2 多智能体Actor-Critic框架计算过程 40](#_Toc56515294)

[4.5 AHAC算法 41](#_Toc56515295)

[4.6 本章小结 43](#_Toc56515296)

[5 第五章 原型系统实现与实验 44](#_Toc56515297)

[5.1 原型系统架构与实现 44](#_Toc56515298)

[5.1.1 环境配置 44](#_Toc56515299)

[5.1.2 原型系统设计 45](#_Toc56515300)

[5.1.2 关键算法实现 47](#_Toc56515301)

[5.2 算法实验与结果分析 51](#_Toc56515302)

[5.2.1 实验基线 51](#_Toc56515303)

[5.2.2 实验环境 53](#_Toc56515304)

[5.2.3 MACTP模型实验与分析 54](#_Toc56515305)

[5.2.4 AHAC算法实验与分析 59](#_Toc56515306)

[5.3 本章小结 65](#_Toc56515307)

[结 束 语 66](#_Toc56515308)

[致 谢 68](#_Toc56515309)

[参考文献 69](#_Toc56515310)

[作者在学期间取得的学术成果 75](#_Toc56515311)

表 目 录

[表3.1 计算引力的奖励反馈值算法 27](#_Toc56515312)

[表3.2 计算斥力的奖励反馈值算法 29](#_Toc56515313)

[表3.3 MACTP模型：基于多智能体Actor-Critic框架的多目标规划模型 30](#_Toc56515314)

[表4.1 基于HAM的信息压缩机制整体流程 38](#_Toc56515315)

[表4.2 AHAC算法：基于分层注意力机制的多智能体强化学习算法 42](#_Toc56515316)

[表5.1 硬件环境详细配置 44](#_Toc56515317)

[表5.2 软件环境详细配置 44](#_Toc56515318)

[表5.3 依赖项详细版本 44](#_Toc56515319)

[表5.4 MACTP模型训练流程 48](#_Toc56515320)

[表5.5 HANCritic类的计算图描述 49](#_Toc56515321)

[表5.6 AHAC算法训练流程 50](#_Toc56515322)

[表5.7 不同的多智能体强化学习算法的对比 52](#_Toc56515323)

[表5.8 神经网络超参数设置 52](#_Toc56515324)

图 目 录

[图1.1 智能体与环境的交互 3](#_Toc56515325)

[图1.2 多智能体环境中智能体与智能体、智能体与环境的交互关系示意图 4](#_Toc56515326)

[图1.3 无人集群多目标规划任务示意图 5](#_Toc56515327)

[图1.4 无人机集群灯光表演 7](#_Toc56515328)

[图2.1 HAN应用实例图，橘色部分代表重点关注的句子，蓝色部分代表重点关注的单词 17](#_Toc56515329)

[图3.1 利用多智能体Actor-Critic框架的MACTP模型训练流程 26](#_Toc56515330)

[图3.2 斥力场作用示意图 28](#_Toc56515331)

[图3.3 MACTP模型架构 30](#_Toc56515332)

[图4.1 AHAC算法整体框架设计 33](#_Toc56515333)

[图4.2 利用分层注意力对信息进行加权压缩 36](#_Toc56515334)

[图4.3 基于HAM的多智能体Actor-Critic框架 40](#_Toc56515335)

[图5.1 原型系统总体架构设计 45](#_Toc56515336)

[图5.2 原型系统UML图 46](#_Toc56515337)

[图5.3 多目标覆盖任务示意图 53](#_Toc56515338)

[图5.4 多追捕者-逃逸者任务示意图 53](#_Toc56515339)

[图5.5 多目标收集任务示意图 53](#_Toc56515340)

[图5.6 多指挥者-行动者任务示意图 53](#_Toc56515341)

[图5.7 多目标覆盖任务中存在3个目标点时的覆盖情况 55](#_Toc56515342)

[图5.8 多目标覆盖任务中存在4个目标点时的覆盖情况 55](#_Toc56515343)

[图5.9 多目标覆盖任务中存在5个目标点时的覆盖情况 55](#_Toc56515344)

[图5.10 存在不同数量目标点时，智能体与其他智能体或者障碍物发生碰撞的总次数 56](#_Toc56515345)

[图5.11 不同数量目标点时，智能体覆盖所有目标点需要的步数 57](#_Toc56515346)

[图5.12 不同数量智能体时，智能体获得的平均全局奖励反馈 58](#_Toc56515347)

[图5.13 不同RF范围时，发生碰撞的次数 59](#_Toc56515348)

[图5.14 不同RF范围时，完成任务的步数 59](#_Toc56515349)

[图5.16 多目标覆盖（3 vs. 3）任务中，智能体获取的平均收益 60](#_Toc56515350)

[图5.17 多追捕者-逃逸者（6 vs. 2）任务中，追捕者获取的平均收益 60](#_Toc56515351)

[图5.18 多目标收集任务中，智能体获取的平均收益 60](#_Toc56515352)

[图5.19 多指挥者-行动者任务中，智能体获取的平均收益 60](#_Toc56515353)

[图5.20 多追捕者-逃逸者（3 vs. 1）任务中，追捕者获取的平均收益 61](#_Toc56515354)

[图5.21 多追捕者-逃逸者（3 vs. 2）任务中，追捕者获取的平均收益 61](#_Toc56515355)

[图5.22 多追捕者-逃逸者（3 vs. 3）任务中，追捕者获取的平均收益 61](#_Toc56515356)

[图5.23 多追捕者-逃逸者（6 vs. 2）任务中，追捕者获取的平均收益 61](#_Toc56515357)

[图5.24 多追捕者-逃逸者（6 vs. 6）任务中，追捕者获取的平均收益 62](#_Toc56515358)

[图5.25 多追捕者-逃逸者（9 vs. 3）任务中，追捕者获取的平均收益 62](#_Toc56515359)

[图5.26 在不同数量的逃逸者的情况下，逃逸者被捕捉的概率 62](#_Toc56515360)

[图5.27 在多追捕者-逃逸者（3 vs. 1）任务中，注意力头结果拼接方式对追捕者策略的影响 63](#_Toc56515361)

[图5.28 在多追捕者-逃逸者（3 vs. 1）任务中，注意力头数量对追捕者策略的影响 64](#_Toc56515362)

[图5.29 在多追捕者-逃逸者（3 vs. 3）任务中，对个体级和组级的注意力权重可视化 65](#_Toc56515363)

摘 要

多目标动态规划是智能无人集群系统协同任务控制中的一个典型问题。多目标动态规划技术可以广泛应用于无人机集群的协同目标跟踪、协同目标打击、协同送货以及协同目标搜索等应用当中，能够为这些应用任务高效完成提供有效支持。多目标动态规划问题最主要的挑战是：如何在一定约束条件下，以最优的方式，在智能体集群中进行动态目标合理分配的同时，规划出所有智能体的无碰撞路径，使得整个任务完成代价最小化。为此，本文针对无人集群多目标动态规划问题展开研究，提出了一个基于多智能体Actor-Critic框架的多目标规划模型以及一种基于分层注意力机制的多智能体强化学习算法，构建了一个基于多智能体强化学习的多目标动态规划原型系统，在此基础上，对本文的研究成果进行了实验验证。本文主要贡献包括以下三个方面：

（1）针对现有方法难以适应未知的动态任务环境以及难以应对任务中智能体数量与目标数量不对等问题，提出了一个基于多智能体Actor-Critic框架的多目标动态规划模型。该模型将智能无人集群抽象为多智能体系统，将多目标动态规划问题抽象为多智能体任务，利用人工势场法设计奖励反馈，同时利用奖励反馈描述任务约束，使用多智能体Actor-Critic框架，在动态环境中对模型进行训练，鼓励智能体充分探索环境，能够使智能体学习到有效的策略，从而提升智能体策略的实时性和鲁棒性。

（2）针对智能体异构且任务中智能体数量增多导致的Critic输入空间维度爆炸问题，提出了一种基于分层注意力机制的多智能体强化学习算法，该算法基于多智能体Actor-Critic框架，采用集中训练、分布执行的模式，利用循环神经网络和分层注意力机制进行信息表示学习和信息加权压缩，提升了算法处理效率，而且该算法既适合静态多目标分配场景也适合动态多目标分配场景，具有良好的适应性和可扩展性。

（3）基于上述研究成果，设计并实现了一个基于多智能体强化学习的多目标动态规划原型系统，搭建实验环境，在OpenAI开源的多智能体粒子环境中构建实验场景，对本文所提出的多目标规划模型和基于分层注意力机制的多智能体强化学习算法进行了实验验证。结果表明，相比于现有的方法，本文所提出的模型和算法在性能、鲁棒性等方面均有明显的提升，并且具有一定的可扩展性。

主题词：智能无人集群；多目标规划；多智能体强化学习；分层注意力机制

ABSTRACT

Multi-target dynamic programming is a typical problem in collaborative task control of intelligent unmanned cluster systems. Multi-target dynamic programming can be widely used in applications such as coordinated target tracking, coordinated target strike, coordinated delivery, and coordinated target search for drone clusters, and can provide effective support for the efficient completion of these application tasks. The main challenge of the multi- target dynamic programming problem is: how to properly allocate dynamic goals in the agent cluster in an optimal way under certain constraints, while planning a collision-free path for all agents to make the entire task Minimize the cost of completion. To this end, the paper carried out research on the multi-target dynamic programming problem of unmanned clusters, and proposed a multi- target planning model based on the multi-agent Actor-Critic framework and a multi-agent reinforcement learning algorithm based on a hierarchical attention mechanism. A multi- target dynamic programming prototype system based on multi-agent reinforcement learning is proposed. On this basis, the research results of this paper are experimentally verified. The main contributions of this paper include the following three aspects:

(1) Aiming at the difficulty of existing methods to adapt to the unknown dynamic task environment and the difficulty of coping with the imbalance between the number of agents and the number of targets in the task, a multi- target dynamic programming model based on the multi-agent Actor-Critic framework is proposed. The model abstracts the intelligent unmanned cluster as a multi-agent system, abstracts the multi- target dynamic programming problem as a multi-agent task, uses the artificial potential field method to design reward feedback, uses reward feedback to describe task constraints, and uses multi-agent reinforcement learning The algorithm trains the model in a dynamic environment, encourages the agent to fully explore the environment, and enables the agent to learn effective strategies, thereby improving the real-time and robustness of the agent's strategy.

(2) Aiming at the dimensional explosion of Critic input space caused by the heterogeneity of agents and the increase in the number of agents in the task, a multi-agent reinforcement learning algorithm based on a hierarchical attention mechanism is proposed, which is based on multi-agent Actor-Critic framework adopts the mode of centralized training and distributed execution, and uses cyclic neural network and hierarchical attention mechanism for information representation learning and information weighted compression, which improves the processing efficiency of the algorithm, and the algorithm is suitable for both static multi-target allocation scenarios and dynamic Multi-target allocation scenarios, with good adaptability and scalability.

(3) Based on the above research results, in the open source multi-agent particle environment of OpenAI, a multi- target dynamic programming prototype system based on multi-agent reinforcement learning was designed and implemented, an experimental environment was built, and an experimental scenario was constructed. The proposed multi- target programming model and the multi-agent reinforcement learning algorithm based on the hierarchical attention mechanism are experimentally verified. The results show that compared with the existing methods, the model and algorithm proposed in this paper have significant improvements in performance and robustness, and they have a certain degree of scalability.

Key Words：Intelligent Unmanned Cluster; Multi-Target Programming; Multi-Agent Reinforcement Learning; Hierarchical Attention Mechanism

# 第一章 绪论

人工智能（Artificial Intelligence，AI）是一种模仿和延伸人类智能以执行任务、基于信息迭代改进的技术[1]。深度学习（Deep Learning，DL）[2]使用多层神经网络提取数据特征，广泛应用于AI领域。强化学习（Reinforcement Learning，RL）的研究目标是解决智能体的决策问题，通过设置合适的奖励值，使得智能体可以通过奖励值反馈，学习到特定的行为[3]。由于在某些场景中，例如Atari游戏，输入数据可能非常复杂，所以将DL与RL相结合，形成深度强化学习（Deep Reinforcement Learning，DRL） [4][5]。智能体可以感知周围环境，自主行动，最大化完成任务概率。最简单的智能体是一个可以解决特定问题的程序，理性、具有思考能力的人类则可以看作是目前最复杂的智能体。智能无人集群系统是指在人工智能的控制下，智能体通过沟通与协作，完成指定的任务。

本文围绕智能无人集群系统群体协同中的多目标动态规划问题展开研究，多目标动态规划任务是智能无人集群系统协同研究中的一个典型的问题，首先，该问题包含两个相互耦合的子问题，即目标分配和路径规划，其核心是要求智能体可以尽量多的捕获环境中的指定目标，并且在行进的过程中对潜在的威胁和其他智能体进行躲避；其次，当目标是具有逃逸策略的智能体时，环境中智能体增多，混合任务中具有多个纳什平衡，无人集群系统中的智能体处理所接收到的外界信息，结合自身的信息进行决策，完成多目标动态规划任务。在本文中，我们使用并改进了多智能体强化学习（Multi-agent Reinforcement Learning，MARL）方法，解决上述智能无人集群系统协同研究中的多目标动态规划问题。首先，针对现有方法计算复杂度高、难以适应未知的动态环境且无法很好地解决当智能体数量与目标数量不对等的问题，我们提出了一个基于多智能体Actor-Critic框架的多目标规划模型，其特点是将目标分配和路径规划的求解有机地结合起来，实现智能体在未知动态环境中进行实时规划；其次，针对环境中存在不同类型智能体且智能体数量较多时导致的维度爆炸问题，我们还提出了一种基于分层注意力机制（Hierarchical Attention Mechanism，HAM）的MARL算法，对智能体接收的信息进行表示学习并对信息进行加权压缩，其特点是可以应用于动态目标的多目标动态规划任务，在实际应用中具有更好的可扩展性；最后，我们在OpenAI开源的多智能体粒子环境[6]中，使用基于多智能体Actor-Critic框架的多目标规划模型，对多目标动态规划任务进行建模，设计奖励反馈描述任务约束条件，结合基于HAM的MARL算法对模型进行训练，设计并实现了一个基于MARL的多目标动态规划原型系统，验证了上述提出的模型和算法的正确性和有效性。

1.1 研究背景和研究意义

1.1.1 研究背景

1956年，正式提出了AI[1]这一术语，属于计算机科学。AI赋予计算机设备学习能力，使其可以像人类一样思考和学习。在当前的研究中，AI面对的难题主要是难以用程序精确描述人类可以很容易解决的任务。90年代，将“理性”与“对象”相结合，完善了“智能体”范式[7]。可以将智能体看做是一个系统，智能体感知周围环境，然后采取动作，智能体的目标是最大化完成任务的概率。

多智能体系统（Multi-Agent System，MAS） 是一组自治的、相互作用的智能体[8]，通过传感器与执行器进行工作。自然界中存在大量的MAS，且MAS中的智能体集体工作时，解决问题的能力和决策上的表现会超越大多数单智能体独立行动时的表现。例如，自然界中食物链顶端的大型个体具有很强的个体能力，但是处于食物链底端的小型个体通过群体行动，也可以造成很大的威慑。智能无人集群的发展思想就是源于这种群体生物学的生存策略，致力于通过数量优势弥补个体能力不足的劣势。对于智能无人集群而言，不仅需要考虑智能体数量的提升，更需要考虑人工智能、群组智能、自动控制等一系列技术，这一技术领域被称为群体智能（Swarm Intelligence）。通过群体智能，MAS中的智能体可以得到最优个体和群体利益[9]，形成智能无人集群。

智能无人集群拥有如下四个方面的特点：一是集群中的智能体通常比较小巧，方便大规模群体行动；二是集群中的智能体数量多，通过数量的优势来保持集群持续工作能力；三是集群内存在异构性，包含多个种类的智能体，可以依据用户需求实现特定任务；第四是集群中的智能体通过智能化的方法进行控制，不关注个体功能，通过大规模部署产生规模效应。智能无人集群通过群体内个体之间的相互协作来展示宏观的智能行为，该行为模式具有四个特点：一是通常采用分布式控制，使集群具有更高的灵活性和鲁棒性；二是集群中任意智能体均能对环境造成影响，且可以感知到造成环境变化的其他智能体；三是采用分治思想完成任务，集群中每个智能体都执行任务的一部分，降低对个体的任务要求；四是群体智能具有复杂性，通过个体交互和协作，达到群体效益大于叠加个体效益的效果。

智能无人集群相较于单智能体具备三点优势：一是集群可以执行更为复杂的任务，降低了续航、载荷能力等因素的影响，通过多智能体协作完成复杂任务；二是集群具有可拓展性，集群规模可以随着任务复杂度的提升进行扩展；第三是集群具有高可靠性，集群中的个体可以相互替代，正常运行的个体可以迅速替代发生故障的个体，继续执行任务，系统具备很强的鲁棒性。

智能无人集群的核心问题是决策问题[10]。机器学习（Machine Learning，ML）是解决AI问题的一个有效方法。通过ML，计算机设备可以从数据中学习出某种概念（或者规律），并且利用学习到的概念（或者规律）去解决特定问题，实现智能。根据研究问题的不同，可以将ML分为三类：监督学习（Supervised Learning，SL）、无监督学习（Unsupervised Learning，UL）和RL[1]。其中，RL的过程可以抽象为智能体从环境中感知当前状态，根据观察值做出相应的行为，环境反馈给智能体相应的奖励，智能体根据环境反馈调整行为，如图1.1所示。



图1.1 智能体与环境的交互

现在的RL基础理论主要包括了三个方面[11]：一是试错法（Trial-and-Error），从实践观察角度研究动物学习过程而得出；二是最优控制（Optimal Control）和动态规划（Dynamic Programming，DP），从控制学角度对智能体的行为进行规划；三是时间差分学习方法（Temporal Difference，TD）[12]，使用后继状态估计当前状态的值函数，是无模型（Model-Free）RL的理论基础。1954年，Minsky首次提出RL的概念和术语[13]。1965年，Waltz等人在控制理论中也提出这一概念，描述通过奖惩的手段进行学习的基本思想[14]。1957年，Bellman提出了求解最优控制问题的马尔可夫决策过程（Markov Decision Processes，MDPs）的DP方法，为研究给定环境中的RL问题提供了一个框架[15]。RL的目标是使得智能体学习到能使设定的长期奖励最大化的策略，即最优策略。随着RL理论和DL技术的进步，组合构成了AI领域的重要技术——DRL[4][5]，并且其应用领域呈现了爆发式增长，例如，自动驾驶、控制论、推荐系统和智能交通网络等等。

RL与规划、MAS相结合，则形成了MARL[16]。MARL将博弈论、RL技术等应用到MAS中，使得智能体能在动态、高维的复杂场景中，与其他智能体进行交互，结合自身的决策，完成多智能体任务[17]，是当前AI领域的研究热点。如图1.2所示，MARL建立一个多智能体共存的任务环境，强调智能体之间的博弈，并研究智能体与环境、以及智能体与其他智能体的交互行为。MARL中的任务场景，根据环境中智能体的任务目标和智能体之间的关系，将任务分为三种类型：完全合作任务、完全竞争任务以及同时存在合作和竞争的混合任务[18]。在完全合作任务中，所有智能体共同合作完成一个任务，属于一个小组；在完全竞争任务中，环境中存在两个或两个以上具有竞争关系的小组，每个小组中仅包含一个智能体；在混合任务中，存在多个具有竞争关系的小组，并且至少一个小组中包含多个具有合作关系的智能体。



图1.2 多智能体环境中智能体与智能体、智能体与环境的交互关系示意图

DL通过对人类神经元的研究，利用多层神经网络强大的数据拟合能力，提取数据特征，是解决输入数据复杂的ML问题的一种有效方法。例如，1998年提出的LeNet-5神经网络结构解决了银行手写数字识别的问题[19]。并且，高性能运算设备的发展解决了DL的计算成本问题，数据量的增多解决了深度神经网络泛化能力不足的问题[9]。DL算法有效地提升了自然语言处理（Natural Language Processing，NLP）[20]、计算机视觉（Computer Vision，CV）[21]等领域的技术水平[22]。将DL应用于MARL，可以有效解决输入数据复杂的问题。因此，多智能体深度强化学习（Multi-agent Deep Reinforcement Learning，MADRL）[16]可以适用于输入数据复杂的多智能体任务，利用DL实现端到端（End-to-End）提取数据特征，之后将提取后的特征输入到MAS模型中，供智能体进行策略的学习。在本文中，所使用的MARL算法均为使用了深度神经网络的MADRL算法。

无人集群多目标动态规划是智能无人集群系统的一个典型任务[23]。如图1.3所示，智能体必须到达环境中规定的每个目标点的位置，并且在向目标行进过程中，同时对环境中存在的障碍物或者威胁，以及无人集群中的其他智能体进行躲避。无人集群多目标规划最主要的挑战就是将集群中的智能体以最优的方式分配给已知的目标，并规划所有智能体的无碰撞路径，并且希望使完成任务的代价最小化，同时尽可能保证智能体的安全。所以，智能无人集群多目标规划任务包含两个相互耦合的子任务。第一个子任务是目标分配（Target Assignment），是指将集群中的多智能体分配到多个目标的过程。这个过程要满足一定的约束条件，通常是所有智能体行进距离最短或者是能量损耗最小，其数学模型属于指派问题[24]，属于NP（Non-deterministic Polynomial）完全问题[25]，通常使用启发式算法进行求解，但仍然具有很高的计算复杂度。第二个子任务是路径规划（Path Planning），是指规划一条完成任务代价最小的路径，智能体沿该路径可以到达指定的目标点，并且需要保证智能体的安全，避免智能体碰撞到集群内的其他智能体、以及环境中的障碍物或者威胁区域。

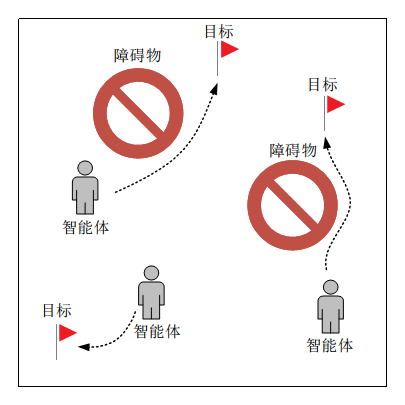


图1.3 无人集群多目标规划任务示意图

智能无人集群的相关技术可以具体应用到无人机集群中，希望无人机集群能代替人类以及有人机更加高效地完成更加复杂的任务。由于无人机集群的应用场景中涉及大量目标分配和路径规划的问题，所以多目标规划任务是智能无人集群研究的重点之一。

1.1.2 研究意义

多目标动态规划任务是智能无人集群的典型任务，是无人机集群、无人车集群遂行任务的基础[23]。智能无人系统中需要自主控制、执行任务，更加依赖对环境的预先建模和实时路径规划，从而可以根据环境变化及时地调整行进路线，规避障碍物和其他智能体，提高效能，增效可靠性。传统的多目标动态规划任务技术存在以下两个方面的问题：一是计算复杂度高，因为目标分配问题属于NP完全问题，可以采用中央控制器的方式来计算分配策略，再发送给智能体，降低个体的计算消耗，但是同时也对智能体间的通信稳定性提出了比较高的要求；二是实时性能差，当无人集群遇到未知的环境或者动态的环境时，提前计算好的目标规划策略可能会失效，需要重新建模并计算新的分配策略并规划路线，将带来较高的延迟。

智能无人集群作为一种典型的MAS，可以将智能体无人集群的多目标动态规划任务看作是一个多智能体任务，根据约束条件，对任务进行建模，使用多智能体Actor-Critic框架训练智能体的策略，使智能体学习如何把当前的观察映射为动作，使数值化的奖励反馈最大化。这类明确考虑了目标导向的智能体与未知的动态环境的交互问题，是目前AI的研究热点之一。在多目标动态规划任务中，设定智能体的目标为到达环境中规定的目标点或者捕获目标点，智能体为了获得更多的收益，必须根据已有的经验获取奖励反馈，同时对环境的未知状态进行探索，追求更好的动作选择空间，通过迭代训练，筛选出最优的策略。

由于在MARL中，智能体不仅需要和环境进行交互，并且需要跟环境中的其他智能体进行交互[9]。因此，不仅智能体本身的决策会对环境产生影响，其他智能体的决策同样会对环境以及智能体自身产生影响。当多智能体任务为混合任务时，即环境中存在不同类型和不同任务目标的智能体，首先判定环境中智能体类别，然后进行智能体之间的交互。当环境中智能体的状态信息属于部分可观测时，即智能体可能无法获取其他智能体的信息，或者智能体无法感知到环境中存在其他智能体，则该环境是动态变化的。因此，环境不具有马尔可夫性，进而无法直接建模为MDPs，所以DRL算法无法直接应用到MADRL中，无法保证智能体策略的收敛性。综上所述，MADRL主要存在以下三个方面的问题：一是当智能体学习策略时，需要建模其他智能体的决策，从而增大了策略学习的复杂度，且在部分可观测情况下不能保证智能体策略收敛；二是智能体之间需要有效地沟通与协作，否则智能体难以学习到最优策略；三是当环境中的智能体之间存在竞争时，则该任务环境中可能存在多个纳什均衡，智能体难以获取一个稳定、明确的策略。

本课题旨在基于MARL算法解决智能无人集群的多目标动态规划问题，发挥MARL的优势，使智能体可以在未知的动态的环境中完成多目标动态规划任务，提高智能体策略的实时性、鲁棒性和可扩展性。同时，我们对现有的MARL算法框架研究了新型MARL算法，从而可以适用于当目标是具有逃逸策略的智能体时的混合任务，提高了算法应用的可扩展性。

1.2 国内外研究现状

1.2.1 智能无人集群

近些年来，智能无人集群的相关技术不管是在理论研究还是工程实践领域中都取得了不俗的成果，也衍生了不同的类型，例如微小卫星集群、无人机集群以及无人车集群等。SpaceX的“星链”计划实现了使用微卫星集群为全球提供互联网服务[26]。智能无人集群的优势之一就是可以通过配备不同类型的智能体，形成不同能力的集群。



图1.4 无人机集群灯光表演

无人机集群是目前国内外研究的热点领域，其中无人机集群灯光秀是民用领域的一个典型应用[27]，如图1.4所示，在工程性和展示性上日趋成熟的无人集群队形变换演示，其无人机集群的规模可以达到上千架，它在地面系统的控制下进行编队变换。但是，其控制原理是集中式控制，即由地面的服务器统一负责调度和控制，无人机端负责执行命令。根据图案的设计，地面控制器预先制定每架无人机的飞行路径和最终位置，然后无人机统一起飞演示，属于一种预规划的自动化行为，无人机端不具有智能性。随着智能无人集群技术的发展，无人机集群的目标是可以通过环境信息和自身信息，在无人机端进行自主决策。

在军事领域内，群体智能作为一种重要颠覆性技术，一直被作为军用人工智能的核心[28]。MIT的“山鹑”无人机实现了在发射后的短时间内与队友交互并组成集群[29]。DARPA的“小精灵”项目可以在防御区外发射具备组网与协同功能的无人机集群，完成离岸侦察等任务，且幸存无人机在任务完成后可被回收[30]。低成本无人机集群技术项目（Low-cost UAV Swarming Technology，LOCUST）中具有数据共享、自主协同能力的无人机群可快速启动，且实现自主集群飞行、协同配合，完成对敌侦察和打击等任务[31]。2017年，中国电子科技集团进行119架固定翼无人机集群飞行试验，完成了多目标分组、合作行动等动作[27]。

1.2.2 多目标规划问题

在目前的智能无人集群技术研究中，多个同构或者异构的智能体协同覆盖或者捕获多个指定目标的多目标规划问题是研究热点之一，主要包含两个子任务：一是目标分配，即满足一定约束条件的指派问题[24]；二是路径规划，规划到达目标的最短路径，并且避免碰撞其他智能体或者障碍物，即避撞。

目标分配是指根据既定的任务，把需要覆盖（或捕获）的目标合理地分配给无人集群系统中的个体，达到高效率执行和高质量完成任务、优化无人集群系统的目的[32]。该问题在军事运筹领域具体为武器-目标分配[33]。目标分配的研究方向可以分为模型建立的研究和模型求解的研究。在模型建立的研究中，研究内容主要包含模型假设、选择目标函数、确定约束条件以及考虑任务代价等多个方面。在模型求解的研究中，主要分为三类：一是传统算法，主要包含整数规划（Mixed Integer Programming，MIP）等；二是启发式算法，主要包含遗传算法（Genetic Algorithm）等；三是将传统算法和启发式算法相结合。例如，文献[34]应用分支界定法，解决多个处理机的分配调度问题；文献[35]基于分工机制的蚁群算法，对无人机协同多任务分配模型进行求解；基于信念点的策略生成算法（Point-Based Policy Generation，PBPG）[36]利用基于分布式部分可观察马尔科夫决策过程（Decentralized Partially Observable Markov Decision Processes，DEC-POMDPs）的多智能体决策模型，对目标分配问题进行建模，根据信念点生成最优策略。

路径规划是指在环境中设计一条满足一定约束的路线，且使一个或者多个确定的目标最优化的问题，包括著名的车辆路径规划问题（Vehicle Routing Problem，VRP）[37]和旅行商问题（Traveling Salesman Problem，TSP）[38]等，可以应用到应急疏散路线设计、救援物资输送路线设计以及物流配送优化等诸多领域，具有广泛的应用前景[39]。在传统的路径规划算法中，主要的解决方法可以分为两大类：精确算法和启发式算法。第一类精确算法是指利用DP、MIP等数学规划方法解决问题。例如，基于免疫亲和模型的紧急救援调度模型[40]受免疫系统的启发，提出了一种基于MIP的集装箱供应链应急救援路径选择模型；基于DP的分解算法[41]将路径规划问题定义为在多模态时间依赖网络中的多准则最短路径行程规划问题，设计分解方案，将问题解耦为多个子问题并利用DP方法求解。由于精确算法的计算复杂度对问题规模敏感，所以精确算法适用于小规模路径规划问题[39]。第二类启发式算法（Heuristic Algorithms）中主要包含了遗传算法、粒子群算法（Particle Swarm Optimization）以及蚁群算法（Ant Colony Algorithm）等。例如，针对VRP问题，混合多目标进化算法（Hybrid Multi-Objective Evolutionary Algorithm，HMOEA）[42]设计专用的遗传算子，结合表示法和局部搜索启发式方法，解决路线日常安排的多目标路径规划问题；针对TSP问题，文化基因随机密钥遗传算法（Memetic Random-Key Genetic Algorithm）[43]用模态随机密钥遗传算法来寻找对称多旅行商问题的最优解。

目前，有相关工作尝试将目标分配和路径规划同时完成，从而可以实现多目标动态规划。例如，目标分配和规划算法（Goal Assignment and Planning，GAP）[44] 解耦目标分配和路径规划，将路径规划问题视为为目标寻找最优分配的目标分配问题的一种扩展，为多智能体生成动态可行的路径，导航到指定的目标。目标分配与路径规划方法（Simultaneous Target Assignment and Path Planning，STAPP）[45]利用MARL算法对智能体的策略进行训练，设计奖励反馈函数，鼓励智能体前往指定的目标点，并指导智能体在行进的过程中远离潜在的威胁，同时解决智能体的目标分配与路径规划问题，为智能无人集群的多目标动态规划问题提供了新的研究思路和研究方向。

1.2.3 多智能体深度强化学习

1.2.3.1 深度强化学习

经典的RL算法Q-learning[46]利用自举法更新贝尔曼方程以及Q表内的Q值。但是在现实世界中，智能体很难在未知、动态的环境中直接计算状态转移概率[47]。因此，可以使用一个函数来逼近状态值（称为值函数）或策略（称为策略函数），该函数可以是简单的线性回归模型，也可以是深度神经网络。

在利用函数逼近器求解最优策略时，目前主要存在两类方法：一是基于价值函数的方法，即学习一个价值函数或状态-价值，用于计算接近长期收益的期望总和。例如深度Q网络（Deep Q-Network，DQN）[4][5]，深度循环Q网络（Deep Recurrent Q-learning，DRQN）[47]等；二是策略逼近方法，即直接学习策略，策略决定了智能体在给定状态下选择行动的概率。例如，使用蒙特卡洛（Monte Carlo）方法的REINFORCE算法[49]，策略梯度算法（Policy Gradient，PG）[50]、对REINFORCE进行扩展的演员-评论家框架（Actor-Critic，AC）[51]，以及运行多个Actor-Critic模型实例并异步更新主节点参数的异步优势Actor-Critic算法（Asynchronous Advantage Actor Critic，A3C）[52]等。

随着DL的快速发展，深度神经网络强大的函数逼近能力可以帮助无模型（Model-Free）RL中的智能体学习准确的值函数或者策略，成为了替代人工指定特征或规则的最佳手段，并且为性能更好的端到端模型的实现提供了可能。DRL目前在部分领域中取得了超越人类顶级专家的水平，例如DQN在Atari游戏中取得了优异的表现[5]，成为了当前AI领域的研究热点。

1.2.3.3 多智能体强化学习

受DRL成功的影响，自2016年开始，MADRL得到了快速的发展，主要分为两种类型，即集中式RL方法和分布式RL方法[53]。集中式RL方法实际上是将一个MAS看为一个具有中央控制单元的单智能体系统（Single-agent System，SAS），例如典型的Central-V算法[54]。集中式RL方法中的每个智能体依靠中央控制单元统一进行学习和调度。虽然集中式RL方法可以使用单智能体的RL算法，有较好的收敛效果和理论保证，但是需要统一对所有智能体的信息进行收集，效率较低，难以付诸实际应用，而且容易出现维数灾难问题。分布式RL方法的每个智能体拥有决策能力，每个智能体可以根据自身观察到的环境状态自主决策。在分布式RL方法中，可以根据系统中的智能体进行决策时是否考虑其他智能体的动作，分为独立式RL方法和协同式RL方法[53]。独立式RL方法中的多个单智能体相互独立，环境中的智能体不考虑与其他智能体的交互，每个智能体使用单智能体RL算法；由于在训练过程中智能体不断改变策略，对每个智能体而言环境都是非平稳的，无法满足单智能体RL算法需遵循的MDPs，所以独立式RL方法不能保证模型收敛。协作式RL方法与独立式RL方法相比，智能体间加入了协调合作机制，让多智能体能够在决策前考虑智能体之间的相互影响，智能体的策略最终满足合理性和收敛性，是目前主流的MARL方法。

MADRL目前主要存在四个主流研究方向[18]。一是智能体行为分析研究。该研究方向的工作主要是将现有的DRL算法，例如将DQN[5]、近端策略优化算法（Proximal Policy Optimization，PPO）[55]等应用到多智能体环境中，对智能体在完全合作任务、完全竞争任务和混合任务中的行为进行分析和评估[56] [57]。二是智能体间的通信协议研究。在该研究方向中，智能体与智能体之间的一种交互方式是通过通信交流信息，最大化共享效益，帮助智能体学习策略。例如，CommNet[58]允许在每个时间步上进行多个通信循环，并且允许环境中的智能体的数量发生动态变化；多智能体双向通信网络（Multiagent Bidirectionally Coordinated Network，BiCNet）[59]提出了使用双向循环神经网络（Bidirectional Recurrent Neural Network，BRNN）[60]来实现多智能体Actor-Critic框架，智能体通过隐藏层信息进行通信，使用参数共享实现信息交互。三是学习合作策略研究。在该研究方向中，智能体间不涉及通信的设置，而是将传统的MAS中有关智能体合作的工作和概念，与DRL的工作相结合，扩展到MADRL任务中。例如，多智能体深度确定性策略梯度算法（Multi-Agent Deep Deterministic Policy Gradient，MADDPG）[61]采用了DRL算法DQN中的经验回放池[4]；多智能体反事实基线算法（Counterfactual Multi-Agent，COMA）[54]使用集中式Critic网络和反事实优势函数，结合了深度确定性策略梯度算法（Deep Deterministic Policy Gradient，DDPG）[62]。四是对其他智能体进行建模研究。在该研究方向中，智能体对其他智能体的策略进行建模，推断其他智能体行为[63]。例如，深度强化对手网络（Deep Reinforce Opponent Network，DRON）[64]是最早提出利用深度神经网络对智能体进行建模的研究工作；基于多智能体Actor-Critic框架的MADDPG算法[61]利用集中式Critic网络，结合其他智能体的信息对自身的行为进行评估，帮助自身的Actor网络学习策略。极大极小多智能体深度确定策略梯度算法（Minimax Multi-agent Deep Deterministic Policy Gradient，M3DDPG）[65]对MADDPG算法进行了扩展，从鲁棒强化学习（Robust Reinforcement Leaning）[66]中引入极大极小思想，假设环境中的其他智能体都会对自身产生负面影响，利用“最坏噪声”（Worst Noise）[67]提升智能体的鲁棒性。

随着MADRL理论和算法的快速发展，MADRL的相关技术和研究成果已经成功应用到人机博弈[68]-[69]，机器人系统[70]以及自动驾驶[71]等诸多领域当中，在这些领域中发挥着重要作用。

1.3 研究目标及研究内容

1.3.1 研究目标

本文以智能无人集群的多目标动态规划任务为背景，研究探索一种MARL的多目标动态规划技术，发挥MARL在解决智能体与未知环境交互问题上的优势，解决现有的多目标规划工作在未知的动态环境中的实时性问题，为解决多目标动态规划问题提供有效支持。首先，针对现有方法难以满足未知的动态任务环境的实时性要求问题，我们对多目标动态规划任务建模进行了研究，设计奖励反馈描述任务的约束条件，鼓励智能体学习到有效的策略，该模型的特点是可以同时解决目标分配和路径规划问题，并且可以适用于智能体数量和目标数量不对等的任务场景；其次，针对动态目标的多目标动态规划任务中Critic输入空间维度爆炸问题，我们研究了如何利用循环神经网络（Recurrent Neural Network，RNN）和HAM对信息进行表示学习和加权压缩，提升多智能体Actor-Critic框架的训练效率，特点是降低Critic网络对智能体数量的敏感性，可以适用于静态目标或动态目标的多目标动态规划任务场景。我们在不同类型的多目标动态规划任务场景下对研究工作进行了测试并进行了实验结果分析，验证了研究工作的正确性和有效性。

本文的研究目标是在调研内容所涉及的方法及其研究思路的基础上，针对现有多目标规划方法难以适应未知的动态环境且难以应对智能体数量与目标数量不对等问题，以及现有多智能体Actor-Critic框架难以应用于动态目标的多目标规划任务问题，我们重点研究了使用基于多智能体Actor-Critic框架的多目标规划模型对多目标动态规划任务进行建模，以及使用基于HAM的MARL算法提高模型训练效果，设计并实现一个基于MARL的多目标动态规划原型系统。并且，我们在MPE场景中设计了多种类型的多目标动态规划任务场景，对我们提出的基于多智能Actor-Critic框架的多目标规划模型（Multi-agent Actor-Critic for Multi Target Programming，MACTP）和基于HAM的MARL算法（Actor Hierarchical Attention Critic，AHAC）进行了实验验证，为MARL算法在多目标动态规划任务中的研究提供了有效支撑，满足了智能无人集群在未知的动态环境中的多目标规划问题所需的实时性、鲁棒性要求，并提升了智能体策略的可扩展性。

1.3.2 研究内容

本文以智能无人集群的多目标动态规划任务为背景，设计并实现了一套集任务建模模块、模型训练模块为一体的智能无人集群多目标动态规划原型系统。通过该系统，可以有效地解决智能无人集群多目标动态规划问题。首先，该系统对智能无人集群多目标规划问题进行建模，转化为多智能体任务，利用人工势场法设计奖励反馈，使用奖励反馈描述任务约束，利用多智能体Actor-Critic框架对模型进行训练；其次，根据实际需要，针对具有逃逸策略的目标，使用HAM对MARL算法进行改进，解决当环境中智能体数量较多时，智能体难以学到有效策略的问题，提高MARL算法的可扩展性。我们主要对以下三个方面进行了探索。

（1）基于多智能Actor-Critic框架的多目标规划模型设计与实现。在多目标动态规划场景中，包含一定数目的智能体、目标点以及障碍物，MACTP模型的主要思想是将智能无人集群抽象为MAS，将多目标动态规划问题抽象为多智能体任务，基于人工势场法设计奖励反馈，并利用奖励反馈描述任务约束条件，使用多智能体Actor-Critic框架训练模型，训练后的模型为智能体策略。同时，为了提升训练后的智能体的鲁棒性和可扩展性，训练过程在动态环境下进行，且鼓励智能体充分探索训练环境中的不同状态。

（2）基于HAM的MARL算法设计与实现。在多目标动态规划任务中，当智能无人集群需要捕获的目标具有逃逸策略时，则该任务则属于混合任务。在混合任务中，无人集群中的智能体在学习策略时，需要同时考虑来自集群中的其他智能体信息和来自待捕获的目标智能体的信息，智能体学习到的策略对接收的信息非常敏感。在多智能体Actor-Critic框架中，随环境中的智能体数量的增多，集中式Critic网络的输入空间维度快速增长，导致Q值的评估变得困难，智能体的Actor网络难以根据Q值反馈学习到有效的策略。针对上述问题，AHAC算法使用多智能体Actor-Critic框架，集中训练、分布执行，利用RNN和HAM进行信息表示学习和信息压缩，不仅可以解决静态目标的多目标规划问题，而且可以解决目标是具有逃逸策略的智能体的多目标规划问题，提高了算法的可扩展性。

（3）基于上述研究成果，设计并实现基于MARL的多目标动态规划原型系统。该原型系统针对传统智能无人集群多目标动态规划算法在动态环境中计算难度大，智能体的策略难以实时应对未知环境的问题，MACTP模型通过构建MAS中的多智能体任务，设计奖励反馈描述多目标动态规划任务的约束条件，利用多智能体Actor-Critic框架训练模型，即训练智能体策略。该原型系统充分发挥了MARL的优势，针对多目标动态规划任务，训练后的智能体可以适应未知的动态环境，根据自身的策略做出决策，提高了智能体策略的实时性和鲁棒性，并且具有一定的可扩展性。并且，针对多智能体Actor-Critic框架中，集中式Critic网络的输入空间维度对智能体数量敏感、难以准确评估Q值的问题，该原型系统采用基于HAM的多智能体Actor-Critic框架，利用HAM对所有智能体的信息进行加权压缩，提高了MARL算法的可扩展性，提升了智能体的性能，不仅可以解决静态目标的多目标规划问题，并且可以适用于目标是具有逃逸策略的智能体的复杂情况。

概括起来，我们将智能体无人集群的多目标动态规划问题建模为MAS中的多智能体任务，设计奖励反馈描述任务的约束条件，并且使用多智能体Actor-Critic框架对模型进行训练；并且根据任务需求，利用HAM对MARL算法进行了改进，提高了MARL算法的可扩展性，推进了MARL算法在智能无人集群中的应用。

1.4 论文的组织结构

本文的内容主要由五个章节及结束语组成，各章节的内容概括如下。

第一章是绪论，首先，论述了本文的研究背景、研究意义及国内外研究现状，主要描述了智能无人集群的定义及应用、多目标规划问题的子问题以及子问题的约束条件、MARL算法的原理和优势，并对现有的工作进行了分析，提出了现有工作需解决的问题；其次，提出了本文的研究目标和研究内容。

第二章是相关工作，对与本文密切相关的问题和技术进行了调研分析，主要包括了奖励反馈设计的意义和在智能无人集群多目标规划任务中的应用；注意力机制的原理和在神经网络中的应用进行了描述；MARL的理论基础、常见的算法分类以及现有的效果较好的方法进行了详细分析和论述，这些技术和方法为我们的研究提供有益的参考，奠定了基础。最后，介绍了原型系统开发和测试所使用的实验平台。

第三章是基于多智能体Actor-Critic框架的多目标规划模型，该模型包含两个核心层，即任务抽象层和模型训练层。在任务抽象层，将智能无人集群的多目标规划任务建模成为多智能体任务，设计奖励反馈描述多目标规划任务的约束条件；在模型训练层，使用多智能体Actor-Critic框架训练模型，保存模型作为智能体的有效策略。

第四章是基于HAM的MARL算法，该算法的实现主要包含了三个机制，即基于RNN的特征提取机制、基于HAM的信息压缩机制和多智能体Actor-Critic框架策略更新机制。该算法使用多智能体Actor-Critic框架，集中训练、分布执行，利用RNN和分HAM进行信息表示学习和信息压缩，解决多智能体混合任务中Critic网络输入空间对智能体数量敏感、无法准确指导智能体学习策略的问题，实现了MARL在动态目标的多目标动态规划问题中的应用。

第五章是基于MARL的多目标动态规划原型系统的实现和实验验证，主要包括了对原型系统架构的设计和实现，并对主要技术，即MACTP模型和AHAC算法，进行了实验验证和性能测试，验证了本文设计的基于MARL的多目标动态规划原型系统相比于现有的多目标规划方法，在鲁棒性、可扩展性等性能上，都取得了一定的提升。

最后是结束语，总结了本文的研究意义、研究内容以及主要贡献，并根据现有的工作，对未来的研究方向进行了展望。

1.5 本章小结

本章首先描述了课题的研究背景和研究意义。根据调研结果，阐述了关于智能无人集群、多目标动态规划问题和国内外MARL的研究现状。其次，阐述了课题的主要研究目标和研究内容是利用MARL方法来解决智能无人集群的多目标规划问题，将该问题建模为多智能体任务，使用奖励反馈设计描述多目标规划问题的约束条件，使智能体可以同时动态完成目标分配和路径规划，并针对实际情况，考虑了当智能体数量和目标数量不对等的非对称场景；针对目标是具有逃逸策略是智能体的混合任务，对现有的MARL算法进行改进，引入HAM，增强MARL算法的可扩展性。最后，概述论文组织结构。

# 第二章 相关工作

本章将详细论述和分析了与本文的研究内容相关的关键技术和工作，主要包括对智能无人集群多目标规划问题的调研和分析，对奖励反馈设计的调研和分析、对注意力机制（Attention Mechanism，AM）的应用的调研和分析以及对目前效果较好的MARL方法的调研和分析。

2.1 奖励反馈设计

在RL中，智能体的任务目标被形式化表征为一种特殊信号，称为奖励反馈（或者收益）[11]。智能体根据当前所处的状态执行动作后，环境反馈后继状态和收益。在每个时间步，奖励反馈都是一个单一的标量数值。对智能体而言，它的目标是最大化其收到的所有奖励。这意味着智能体需要最大化的不是收到的当前奖励，而是一个规定时间窗口内的长期的累积奖励。研究人员将这种想法表述为奖励假设：所有的研究的目标都可以归结为最大化智能体接收到的标量信号（即，奖励反馈）累积和的概率期望值[11]。智能体使用奖励反馈来形式化其目标是RL最显著的特征之一。

当前RL面临的一个调整是奖励反馈稀疏问题，即当智能体需要做关键性动作来完成复杂任务或者一系列任务的时候，需要足够频繁地提供非零收益让智能体可以实现目标，但是可以明确触发奖励反馈的“状态-动作”二元组可能很少并且可能间隔很远，智能体可能会长时间没有目的地漫游，难以学到策略。解决稀疏奖励反馈问题的一个手段是奖励设计（Reward Shaping），引入人工先验知识，对智能体探索环境所获得的奖励反馈进行加工设计，以帮助提升其学习效率，使其策略尽快收敛。

吴恩达等人[72]在提出了在设计奖励反馈时，保证最优策略不变的条件是当附加奖励反馈可以表示为任意一个状态的势能函数（Potential-Based Function），势能函数被定义为状态到实数的映射的差分形式。同时，该研究工作设计了基于距离和基于子目标的启发式奖励函数，并通过实验证明了奖励反馈设计可以帮助缩减智能体的训练时间。

在线循环确定策略梯度算法（Fast Recurrent Deterministic Policy Gradient，Fast-RDPG）[73]将问题描述为一个部分可观察马尔科夫决策过程（Partially Observable Markov Decision Processes，POMDPs），无人机的策略通过DRL算法训练，可以在复杂环境中执行导航任务。与传统的路径规划方法相比，Fast-RDPG直接将无人机原始观察值映射为导航控制信息，使无人机可以在未知复杂场景中，根据自身的观测值自主导航。但是Fast-RDPG关注于单智能体任务，当复杂环境中存在多个个体时，存在多个个体之间难以进行协调控制的问题。

STAPP方法[45]将MARL算法应用到多目标动态规划任务中，设计奖励反馈函数，训练智能体的策略，实现同时解决目标分配问题与路径规划问题。与传统集群无人机的多目标规划算法相比，不需根据任务目标和特定约束条件建立数学模型，适用范围更广。但是，STAPP方法仅考虑了当无人机与目标数量相等，且无人机数量较少（智能体数量不多于5个）的情况。当无人机小于目标数量时，存在无人机无法到达全部目标点的问题；当无人机和目标的数量过多时，存在学习策略过慢的问题。

2.2 注意力机制

AM最早应用于CV[74]，已经成为神经网络中的一个重要概念，在不同的应用领域得到了广泛的研究[75]。我们可以参考人类的生物系统，对注意力机制进行解释。例如，我们的视觉系统倾向于选择性地关注视线中景象的一部分信息，忽略其他不相关的信息，从而有助于感知和处理信息。以此类推，在涉及CV、NLP等任务中，输入信息的某一部分与其他部分相比，与需要完成的任务可能更为相关。例如，在文档翻译的任务中，输入序列中的某些关键单词与预测下一个单词的含义相关。AM允许模型动态地关注输入信息的某些部分，结合相关性的计算和概念，有助于有效地执行当前的任务[75]。AM具有三点优势：第一，很多应用了AM的神经网络模型是目前性能比较好的模型，可以用于机器翻译、对话系统等多项任务，具有广泛的应用前景；第二，AM可以增加神经网络的可解释性；第三，AM可以解决目前神经网络存在的一些问题，例如神经网络的计算效率随输入信息维度的增多而下降。

软注意力机制（Soft-Attention）[76] 输入的信息序列的不同的部分赋予不同的权重，上下文向量由信息序列的所有隐藏状态加权计算得到，实现对信息进行软区分的目的。将源信息转化为一系列的“键值（Key，）-价值（Value，）”数据对，给定目标信息中的某个查询值（Query，），通过计算查询值和源信息中各个键值的相似性或者关联程度，得到每个键值对应的价值的权重系数，对价值进行加权求和，计算得到目标信息中某个元素的最终结果。与硬注意力机制（Hard-Attention）[77]是一个随机的过程不同，Soft-Attention是参数化的，可以被嵌入到网络模型中，使得神经网络能够通过梯度的反向传播进行有效的学习。

2017年，Google提出的Transformer[78]中将自注意力机制（Self-Attention）和多头注意力机制（Multi-Head Attention）相结合，在序列到序列（Sequence to Sequence，Seq2Seq）问题中，使用全AM结构，明显提升翻译任务中的训练效果。在Seq2Seq问题中，输入的源信息和输出的目标信息的内容是不一样的，AM通常应用在目标信息的Query和源信息的所有Key之间的对应的计算中。但是，Self-Attention是应用在源信息或者目标信息的内部，Query、Key和Value均来自于同一个输入序列。使用缩放因子的缩放点乘注意力机制（Scaled Dot-Product Attention）如式（2.1）所示。Transformer证明Self-Attention可以捕获长距离的相互依赖的特征，从而可以学习同一个句子中的某些句法特征和语义特征[78]。

|  |  |
| --- | --- |
|  | （2.1） |

在Multi-Head Attention中，将源信息分成个部分，将每个部分映射到一个独立的特征子空间（即，注意力头），在该注意力头内使用Scaled Dot-Product Attention进行计算，最后将多个注意力头的计算结果进行拼接，如式（2.2）所示。

|  |  |
| --- | --- |
|  | （2.2） |

其中，代表注意力头的数量，代表第个注意力头的计算结果，代表用于数据映射的可学习的参数矩阵。

分层注意力网络（Hierarchical Attention Networks，HAN）[79]应用于自然语言处理中的文档分类任务。HAN包含两层注意力计算，分别是单词级注意力和句级注意力。HAN利用这种分层架构，首先通过单词级注意力计算句子的价值，然后通过句级注意力计算文档层级（或段落层级）的语义向量，用于文档的情感分类计算。HAN的应用实例如图2.1所示，该实例包含5个句子，由句号、问号分隔。由语义可知，第一句和第三句表达情感更强烈，其中，delicious这个词最能表现这段话的情感。

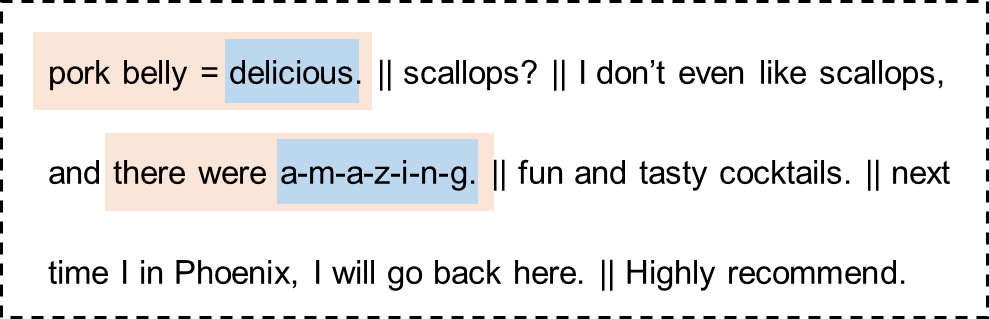


图2.1 HAN应用实例图，橘色部分代表重点关注的句子，蓝色部分代表重点关注的单词

2.3 多智能体强化学习

2.3.1 马尔科夫博弈

本文的研究内容使用马尔科夫博弈（Markov Game）框架，该框架基于多智能体MDPs[80]，可以使用一个元组来表示。其中，是环境中智能体的数量；是有限状态集合；是有限动作集合，是状态观察值集合；给定智能体当前状态和执行动作，是智能体的状态转移函数，定义智能体的后继状态概率分布；是根据目标设置的智能体的奖励反馈函数，依赖全局状态和动作。在POMDPs中，智能体的观察值包含全局状态的部分信息；初始状态由分布决定；每个智能体学习一个策略，将自身的观察值映射到其动作空间上的分布。智能体的学习目标是最大化期待奖励反馈，如式（2.3）所示。其中，是未来奖励反馈的折扣因子，是时间窗口。

|  |  |  |
| --- | --- | --- |
|  |  | （2.3） |

将从状态开始执行策略π的价值函数设为，如式（2.4）所示。

|  |  |  |
| --- | --- | --- |
|  |  | （2.4） |

在给定状态转移概率和奖励反馈时，式（2.5）适用于在任何时间步时的全部状态。

|  |  |
| --- | --- |
|  | （.5） |

其中，是的后继状态。基于策略的RL算法，可以通过式（2.6）获得最优策略，选取使设定奖励最大的动作。

|  |  |
| --- | --- |
|  | （.6） |

同理，基于价值的RL算法可以通过式（2.7）计算各个智能体的状态-动作函数的最优Q值。

|  |  |
| --- | --- |
|  | （.7） |

2.3.2 深度强化学习

1989年，Watkins等人提出了Q-learning[46]，其核心思想是利用自举法更新Q表和贝尔曼方程[15]。在训练开始前，随机初始化Q表；在时间步，智能体的策略根据当前状态，选择并执行动作，环境反馈给智能体奖励，并且智能体根据状态转移概率进入后继状态；Q表中的每一项按照式（2.8）进行更新。

|  |  |  |
| --- | --- | --- |
|  |  | （.8） |

其中，为学习率；为奖励的折扣因子，智能体策略的学习兼顾未来收益和当前奖励反馈。每次迭代称为一个训练周期（Episode），回合结束存在两种情况：（1）达到回合规定的最大步数（Step）；（2）智能体完成指定任务。随着训练周期的增加，Q表内的Q值逐渐收敛，代表智能体的策略逐渐稳定，即智能体学习到了一个可以完成任务的可行动作序列。

在RL中，状态维数的增加或者采用连续值状态空间会使函数逼近器的计算量指数级增长，因此可以采用人工先验知识提取特征，以及参数化Q值函数的方法。但是，人工提取特征存在过于依赖先验知识、通用性较差的缺点。DL提供了一种有效的降维方式，通过多层神经元的结构化表征，神经网络可以学到数据有低级特征到高级特征的表达，有效解决维数灾难问题。DRL将DL和RL相结合，实现一种端到端的RL模型。2013年，DQN[5]利用神经网络对评估状态-动作的Q值进行参数化，即计算。同时，结合卷积神经网络（Convolutional Neural Network, CNN），实现了直接从图片输入到动作Q值的映射。DQN定义了如式（2.9）所示的损失函数，最小化该损失函数，更新参数。

|  |  |
| --- | --- |
|  | （.9） |

同时，为了实现异步学习（Off-Policy），提高数据利用率，且减少样本间的相关性，DQN算法提出了用于随机采样的经验回放池[5]，打破了样本间由于环境的马尔科夫性导致的相关性，满足DL样本所需的独立同分布要求。

相对于基于价值的方法，PG方法[50]有两个主要优点：第一是PG方法可以收敛到一个固定的策略，而基于价值的方法一直按照概率选择随机动作；第二是基于价值的方法无法学习到随机策略，但是在竞争环境内，最优策略可能是随机的。PG方法估计有关策略参数的预期收益的梯度，Q值计算如式（2.10）所示。

|  |  |
| --- | --- |
|  | （.10） |

智能体的策略梯度计算如式（2.11）所示。

|  |  |
| --- | --- |
|  | （.11） |

其中，是状态的分布。DDPG算法[62]对确定性策略梯度算法（Deterministic Policy Gradient，DPG）[81]进行扩展，使用经验回放池、并设立目标网络用于保存参数，实现异步策略更新，可以解决连续控制问题的RL问题。

2000年，演员-评论家框架（Actor-Critic）[51]结合深度学习，Actor函数和Critic函数分别由神经网络拟合。其中，Actor网络为策略函数，用于将输入状态映射到动作概率分布[82]；Critic网络为状态值函数，用于计算状态-动作的期望价值，用于指导Actor网络更新。Actor-Critic框架的一个具体实例是学习一个函数，用于估计折扣后的长期奖励反馈，给定一个状态-动作，采用TD方法学习异步策略，最小化如式（2.12）所示的回归损失函数。

|  |  |
| --- | --- |
|  | （.12） |

软演员-评论家框架 （Soft Actor-Critic）[83]鼓励智能体对未知的状态进行探索，通过最大熵，避免收敛到非最优确定性策略。Soft Actor-Critic框架修改智能体的策略梯度，使其包含一个熵项，如式（2.13）所示。

|  |  |
| --- | --- |
|  | （.13） |

并且，Soft Actor-Critic框架对动作-价值函数的损失函数也进行了修正，新的目标函数如式（2.14）所示。

|  |  |
| --- | --- |
|  | （.14） |

2.3.3 多智能体强化学习

MADDPG算法[61]使用集中式Critic网络，将DDPG扩展到MARL方法中。每个智能体包含一个独立的Actor网络和一个集中式Critic网络。集中式Critic网络将所有智能体的状态值和动作值作为输入信息，而 Actor网络只依赖于给定的局部观测信息。MADDPG算法解决了非平稳性问题，并且有效降低了方差。但是，随着智能体数量的增加，MADDPG算法中的集中式Critic网络的输入空间维度快速增长，所以MADDPG算法的可扩展性较差。

COMA算法[54]提出了一种包含反事实基线的集中式Critic网络，用于解决MARL中的信用分配问题。COMA算法中所有智能体共享一个集中式Critic网络，每个智能体包含一个Actor网络。集中式Critic网络利用反事实优势函数进行更新，通过分离特定智能体的行为，该优势函数可以利用深度神经网络有效地计算该智能体的贡献。反事实优势函数鼓励智能体执行对整体奖励影响较大的动作，COMA算法的策略梯度如式（2.15）所示。同时，COMA算法通过边缘化智能体的行为降低方差。

|  |  |
| --- | --- |
|  | （.15） |

其中，代表保持其他智能体的动作不变；为每个智能体计算一个单独的基线，智能体使用集中式Critic网络来推理只有动作值改变的反事实；代表了在状态的所有智能体的联合动作的评估Q值。

基于AM的多智能体Actor-Critic算法（Multi-Actor-Attention-Critic，MAAC）[84]将Soft-Attention应用到集中训练、分布执行的多智能体Actor-Critic框架中。基于AM的Critic网络能够在每个时间步动态地选择要关注部分智能体的信息。MAAC算法通过选择相关信息来提高处理信息的效率，并且通过AM对信息进行压缩，降低了智能体数量增加对Critic网络的输入空间大小的影响，提高了算法的可扩展性。

2.4 其他相关技术

2.4.1 Gym强化学习框架

2016年，OpenAI提供了Gym开源库[85]，可以用于开发RL算法，或者比较已有的RL算法的性能。在Gym标准化的环境中，不对智能体的结构做任何假设。Gym主要支持Python语言，并且与多个常见的数值运算库兼容，例如Tensorflow等。

在RL中，智能体在环境中执行动作，接收环境反馈的后继状态和收益，如图1.1所示。Gym提供了统一的环境接口，智能体的决策由用户实现，可以用于RL算法的开发。Gym提供了多个经典的仿真环境，例如，Atari2600游戏，基于MuJoCo物理引擎的运动控制问题等。同时，作为开源库，Gym允许用户自定义环境和任务，用于设计和实现RL算法。

2.4.2 多智能体粒子环境

2017年，OpenAI开源了一个MARL平台——多智能体粒子环境（Multi-agent Particle Environment，MPE）[6]。MPE是一个具有连续观察值和离散动作空间的简单的多智能体粒子世界，以OpenAI的Gym为基础，包含了基本的模拟物理学。MPE包含了一系列多智能体任务场景，例如，代表完全合作任务的协作导航场景、代表完全竞争任务的追捕场景等。在部分的多智能体任务中，该环境中的智能体允许同时执行物理动作和通信动作，例如，存在交流信息的部分可观察的追捕场景。MPE作为开源的MARL平台，允许用户自定义多智能体环境和任务，便于进行MARL算法的实现和测试。

2.5 本章小结

本章主要对本文所涉及的相关理论和研究工作进行了详细论述和分析，主要包括了奖励反馈设计、AM、MARL以及实验平台。首先，介绍了奖励反馈设计的意义，并简要介绍了在多目标规划任务中的应用。其次，对注意力机制的原理和在神经网络中的应用进行了论述。然后，对本文所使用的主要技术MARL进行了详细论述，包括了理论基础、常见的算法分类以及现有的效果较好的方法，为本文的研究提供参考思路。最后，描述了原型系统开发和实验所使用的MARL算法实验平台。

# 第三章 基于多智能体Actor-Critic框架的多目标规划模型

在多目标动态规划任务中，存在两个相互耦合的子问题：目标分配和路径规划，这两个子问题都可以使用约束条件进行表示。传统的多目标规划算法存在计算复杂、难以满足动态环境的问题。为了解决智能体在未知动态环境中的多目标动态规划问题，提高智能体的鲁棒性和可扩展性，我们提出了基于多智能体Actor-Critic框架的多目标规划模型（Multi-agent Actor-Critic for Multi Target Programming，MACTP）利用奖励反馈设计方法描述多目标动态规划任务的约束条件，使用多智能体Actor-Critic框架对模型进行训练。本章首先对MACTP模型进行整体概述，然后对任务建模和奖励反馈的具体设计方法进行详细阐述。

3.1 MACTP模型概述

智能无人集群的多目标规划问题包含两个相互耦合的子问题，分别是目标分配问题和路径规划问题。其中，路径规划问题包含了两个约束条件：（1）最短飞行距离；（2）智能体避免碰撞其他智能体和障碍物。MACTP模型的主要思想是将智能无人集群抽象为MAS，将多目标动态规划问题抽象为多智能体任务，参考人工势场法设计奖励反馈，利用奖励反馈描述任务约束，使用MARL算法，对任务模型进行训练，使智能体能够学习到有效的策略，满足多目标动态规划任务的实时性要求。同时，为了提升训练后的智能体的策略的鲁棒性和可扩展性，训练过程在动态环境下进行，且鼓励智能体充分探索训练环境中的不同状态。

3.2 多智能体任务建模

智能无人集群的多目标规划问题（Target Programming Problem，TPP）的一般化描述如下：假设智能无人集群内有个智能体，使用对智能体进行索引，智能体需要捕获个目标点，使用对目标点进行索引；每个目标要求能且只能被一个智能体捕获。假设智能体到达目标点的距离（或者是其他代价）为，目标分配向量如式（3.1）所示：

|  |  |
| --- | --- |
|  | （.1） |

目标分配问题实际是求解一个使式子（3.2）成立的策略：

|  |  |
| --- | --- |
|  | （.2） |

在实际应用中，由于需要考虑智能体的动力学要求、避免智能体之间的碰撞以及智能体和障碍物之间的碰撞等问题，不仅需要将集群中的智能体分配给已知的目标点，还需要规划所有智能体的运动路径，使整个任务的完成代价最小化，并且需要使智能体碰撞其他智能体或者障碍物的概率最小化，因此我们对智能无人集群多目标规划任务进一步约束，规定了如下符号，详细定义了这一任务过程。

在多智能体环境中，存在个障碍物，在时间步时，智能体的位置为，目标点的位置为，障碍物的位置为。设定在时间窗口范围内，智能体的运动路径点的集合为，智能体的完成任务的代价为运动路径长度，则智能无人集群多目标规划任务的优化目标为，且必须满足以下四个约束条件：

（1）每一个目标点都要被分配给一个智能体，即；

（2）每一个目标点只能被分配给一个智能体，即；

（3）每个智能体都不能碰撞环境中的障碍物，即在时间窗口内的任意时间步,；

（4）每个智能体不能碰撞集群中的其他智能体，即在时间窗口内的任意时间步，集群中任意两个智能体的位置不能相同，则。

由以上的任务过程描述可知，无人集群多目标规划任务属于多智能体完全合作任务，因此我们可以考虑使用经典的多智能体Actor-Critic框架（例如，MADDPG算法[61]）对模型进行训练，将智能无人集群的多目标规划问题从传统的最优化问题转化为多智能体策略训练问题，从而使无人集群中的智能体可以适应动态环境。

对于环境中的个智能体，所有智能体的策略可以被参数化表示，将所有智能体的策略集合表示为。设智能体的预期奖励反馈为，预期奖励反馈的梯度计算如式（3.3）所示。

|  |  |
| --- | --- |
|  | （.3） |

其中，为智能体的动作值；为智能体的观察值；是智能体集中式的动作-价值函数，输入信息为所有智能体的动作信息和状态信息，输出信息为智能体的估计Q值。在本文中，状态信息设置为所有智能体的观察值，即，每个智能体的观察值中包括如下信息：智能体自身的坐标，智能体与每个目标点之间距离，智能体与集群其它智能体之间的距离以及智能体与障碍物之间的距离。由于每个智能体的都是独立学习的，所以智能体的奖励反馈设计也可以是独立的。

在本文中，所有智能体都采用确定性策略，所以将智能体的策略表示为（缩写为，代表Actor网络），则智能体的预期奖励反馈的梯度如式（3.4）所示。

|  |  |
| --- | --- |
|  | （.4） |

其中，为经验回放池（Experience Replay Buffer），记录每个时间步的所有智能体的经验元组；代表所有智能体在执行动作后，环境的后继状态；是智能体获得的奖励反馈值。集中式动作-价值函数的损失公式如式（3.5）所示。

|  |  |
| --- | --- |
|  | （.5） |

其中，代表目标Actor网络集合，智能体的目标Actor网络使用延迟参数。

通过构造多智能体任务模型，利用多智能体Actor-Critic框架（例如，MADDPG算法[61]），使智能体通过探索环境学习策略，可以同时解决无人集群多目标规划任务中相互耦合的目标分配和路径规划问题。训练MACTP模型的流程如图3.1所示。首先，初始化环境以及随机初始化网络参数，在时间步，智能体的Actor网络根据输入的观察值输出动作值，智能体执行动作，环境反馈后继状态的观察值以及奖励反馈值；将时间内的所有智能体的联合经验存入经验回放池；当经验回放池内存入足够多的经验时，开始对Actor-Critic框架进行训练。其次，从经验回放池内抽取S条经验，将下一时间步的状态信息中的观察值输入智能体的目标Actor网络，得到智能体根据当前策略，对下一个待执行的动作的预判，所有智能体在下一个时间步的待执行的联合动作值为，将状态信息和联合动作值输入Critic网络，计算，将下一时刻的状态信息和预判的联合动作值输入目标Critic网络，结合奖励反馈，计算智能体的预期长期收益，结合式（3.5），对Critic网络进行更新。然后，智能体根据Critic网络反馈的估计Q值，结合式（3.4），对Actor网络进行更新，更新后的Actor网络根据观察值生成新的动作值，执行所有智能体的新的联合动作，得到新的下一时间步的状态信息和奖励反馈，作为新的经验存入经验回放池。最后，利用如式（3.6）所示的软更新方式，更新目标Actor网络和目标Critic网络的参数。重复该训练过程，直到达到规定的最大训练周期数。

|  |  |
| --- | --- |
|  | （.6） |

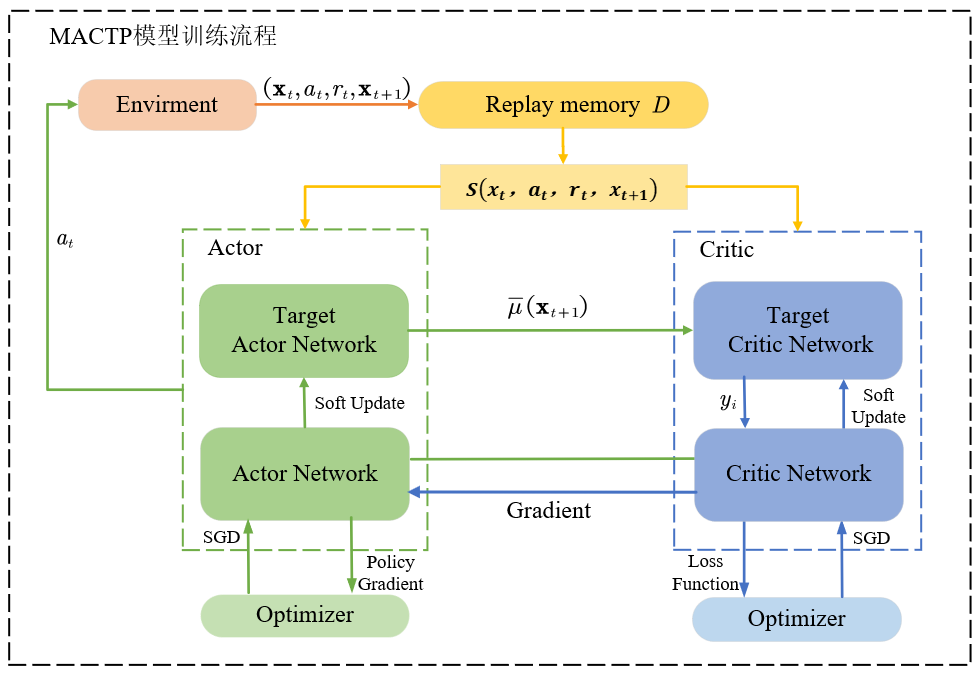


图3.1 利用多智能体Actor-Critic框架的MACTP模型训练流程

3.3 基于人工势场法的奖励反馈设计

在RL中，设定的智能体需要学习如何最大化奖励反馈，所以提供获取奖励反馈的方式必须要使得智能体在最大化自身奖励的同时，也可以实现其设定的任务。因此，设计的奖励反馈要能表明需要完成的目标。值得注意的是，奖励反馈并不单纯是依靠实现目标的先验知识来传授智能体如何完成任务。例如，在国际象棋中，我们通常设定智能体只有当最终获胜时才能获得奖励反馈，而并非达到某个子目标时就可获得奖励反馈（例如，将子目标设定为吃掉对方的棋子），属于稀疏收益。因为子目标可能使智能体对任务的理解产生偏差，假设智能体发现仅实现子目标即可获得收益，那么智能体可能会放弃实现真正的目标，选择实现子目标的方式获取奖励反馈（例如，智能体可能会找到一种以输掉比赛为代价的方式来吃对方的棋子），最终无法学习正确策略。所以，奖励反馈只能用来传达什么是我们想要实现的目标，而不是如何实现这个目标。相关工作已经表明，将启发式知识引入奖励反馈设计可以提升训练效果[72]。为此，我们设计了一种非稀疏的奖励反馈，结合了人工势场法，同时保证智能体的策略不变性。非稀疏的奖励反馈主要包含以下两个部分，即引力反馈值和斥力反馈值，引力反馈值鼓励智能体向目标点前进，斥力反馈值警告智能体远离其他智能体和障碍物，智能体受引力反馈和斥力反馈共同影响，有效、合理地规划路径，下面分别进行详细论述。

3.3.1 引力反馈设计

引力部分的奖励反馈设计主要是鼓励智能体寻找最短的运动路径。在实际应用中，智能体的续航是有限的，尽可能地减少智能体的运动路径距离，可以提高整体任务的完成概率。

表3.1 计算引力的奖励反馈值算法

|  |
| --- |
| 引力的奖励反馈值的计算方法 |
| 1. Initialize 2. **for** target to **do**: 3. **if** = false **then**： 4. Calculate the distance set  of all agents () to the target 6. **if**  **then**: 7. = true 8. **end if** 9. **end if** 10. **end for** 11. **return** |

智能体受到的引力的奖励反馈值的计算方法如表3.1所示。其中，为一个预定义的正常数，用于增加距离的影响；第1行中，初始化引力的奖励反馈值为0；第2行到第10行中，遍历所有目标点，针对目标点，为目标点的捕获标记，用于判断该目标点是否已经被智能体捕获，当目标点还未被捕获时，则计算所有智能体到目标点的距离，构成距离集合，选取中的最小值，取其相反数，扩大倍，累加到中；第6行到第7行中，，为智能体的大小，为目标点的大小，当最小值小于时，则代表目标点已经被捕获，设置为true。在多智能体Actor-Critic框架（例如，MADDPG算法[61]）中，智能体以最大化奖励反馈值为目标，目标分配问题以最小化运动距离为训练目标，故通过将取距离目标的最短距离的相反数，使优化目标得到统一。经过多次迭代训练以后，智能体将在每次行动中选择使总体运动距离最短的目标，即完成了目标分配过程。

3.3.2 斥力反馈设计

斥力部分的奖励反馈设计主要是保证智能体在完成任务过程中，避免碰撞其他智能体或者障碍物。在实际应用中，集群中的多智能体可能会在相近的区域执行任务，或者智能体可能会在障碍物周边执行任务，此时需要考虑避撞问题。当智能体远离其他智能体或者障碍物时，只需考虑快速靠近目标，完成任务即可。因此，我们设置了斥力惩罚的作用范围，称其为斥力场（Repulsive Field，RF），如图3.2所示。

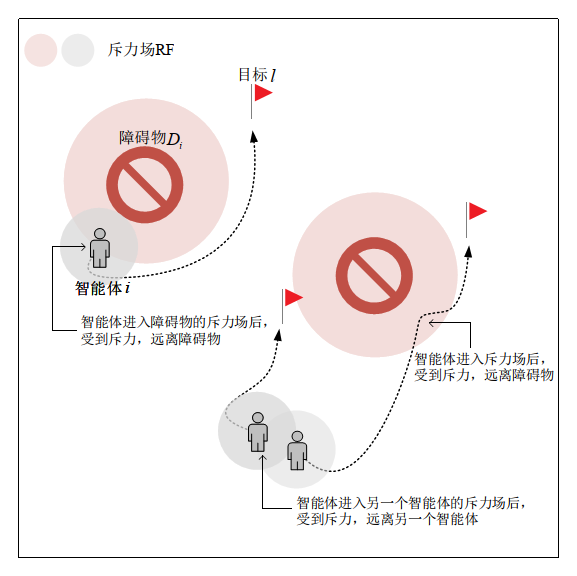


图3.2 斥力场作用示意图

通过设定斥力场，通过斥力反馈，提前对智能体进行预警，当智能体进入其他智能体或者障碍物的斥力场时，智能体不仅需要考虑目标的引力反馈，并且需要考虑斥力场内的斥力反馈；在斥力场外，智能体则可以只考虑目标提供的引力反馈。在斥力场内，斥力反馈的大小与智能体和其他智能体或者障碍物之间的距离有关，距离越近，斥力越大，即智能体越接近其他智能体或者障碍物，它受到的斥力惩罚越大。当智能体与其他智能体或者障碍物碰撞时，则会受到一个较大的惩罚。经过多次迭代训练以后，可以有效的减少智能体之间碰撞的概率，从而解决路径规划中的智能体避撞问题，提升实际应用中任务的完成概率。

表3.2 计算斥力的奖励反馈值算法

|  |
| --- |
| 斥力的奖励反馈值的计算方法 |
| 1. Initialize 2. **for** agent to **do**: 3. **for** agent to **do:** 4. Calculate the distance  between agent and agent 5. **if**  **then**: 7. **else if**  **then**： 9. **end if** 10. **end for** 11. **for** obstacle to **do**： 12. Calculate the distance  between agent and obstacle 13. **if**  **then**: 15. **else if**  **then**： 17. **end if** 18. **end for** 19. **end for** 20. **return** |

智能体受到的斥力的奖励反馈值的计算方法如表3.2所示。其中，RF为预定义的正常数，即斥力场的范围；和为预定义的正常数，用于增加距离的影响；第1行中，初始化斥力的奖励反馈值为0；第2行到第19行，遍历所有智能体，针对智能体，在第4行到第10行中，计算智能体是否可能碰撞到其他智能体，设置，和分别为智能体和智能体的尺寸，当智能体之间发生碰撞时，则为智能体累加一个碰撞惩罚到，当智能体进入智能体的斥力场但是未发生碰撞时，则累加一个基于距离的惩罚警告，为距离放大系数；在第11行到第18行中，同理计算智能体是否可能碰撞到障碍物，，为障碍物的尺寸，设置碰撞惩罚为，距离放大系数为。

3.4 MACTP模型

MACTP模型包含两个核心层，具体架构如图3.3所示。

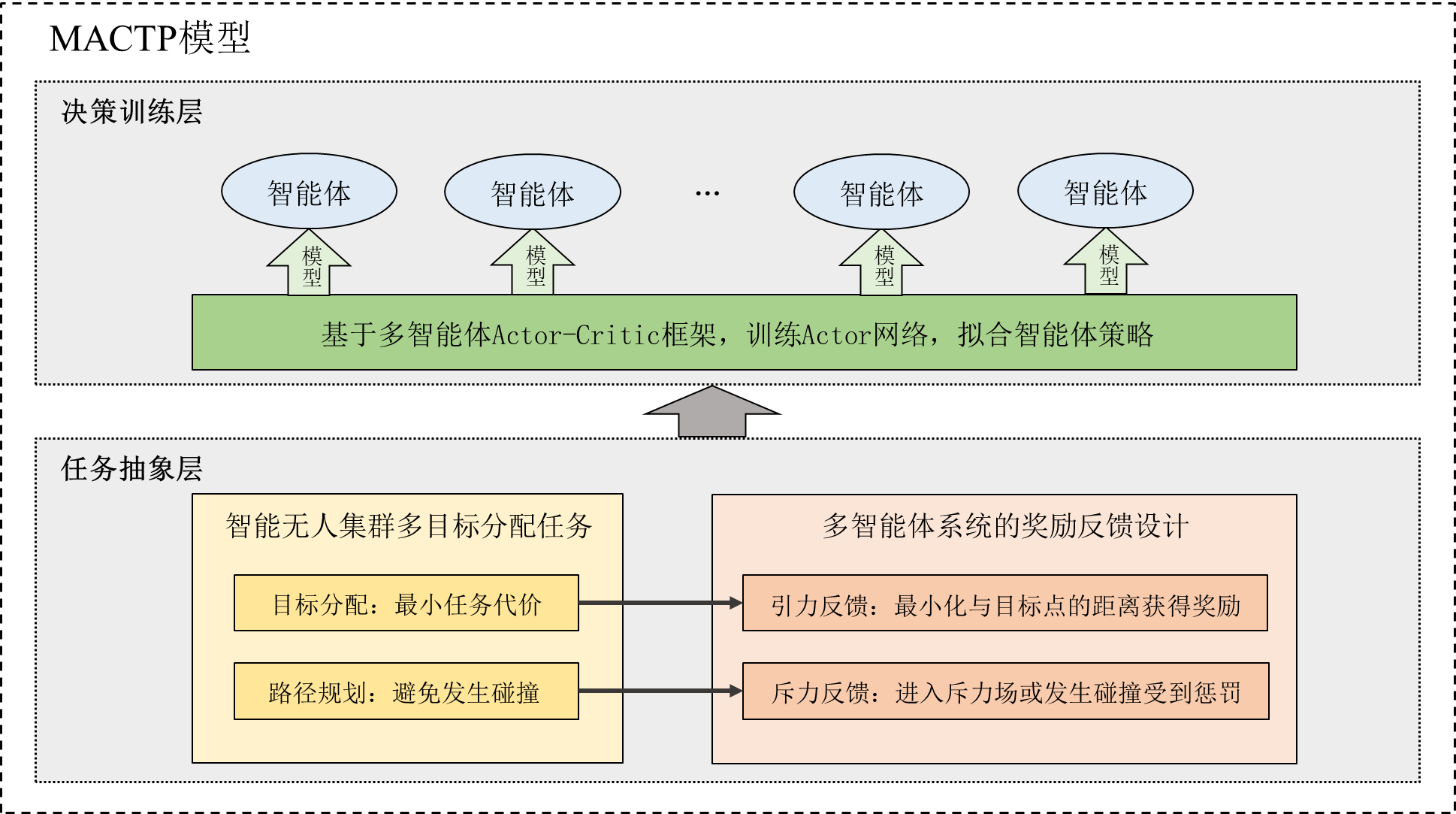


图3.3 MACTP模型架构

第一层是任务抽象层，在该层级内，MACTP模型将智能无人集群的多目标规划任务建模成为多智能体任务，将无人集群多目标规划任务的最优化过程的约束条件，转化为对应多智能体任务中可以用于训练智能体策略的奖励反馈规则，具体规则设计如表3.1和表3.2所示。

第二层是策略训练层，在该层级内，MACTP模型结合多智能体Actor-Critic框架（例如，MADDPG算法[61]），规定智能体策略的输入信息和可执行动作，利用Actor网络拟合智能体的策略；同时，结合任务抽象层，对智能体进行训练，训练过程如图3.1所示，保存模型，即智能体的策略。使用多智能体Actor-Critic框架训练MACTP模型的伪代码如表3.3所示。

表3.3 MACTP模型：基于多智能体Actor-Critic框架的多目标规划模型

|  |
| --- |
| 使用多智能体Actor-Critic框架训练MACTP模型 |
| 1. Initialize the environment with agents, targets and obstacles 2. **for** episode=1 to M **do:** 3. Initialize a random process N for action exploration 4. Receive initial state x 5. **for** to max-episode-length **do:** 6. for each agent , select action 7. Execute actions  and observe reward and new state 8. Store  in replay buffer D 10. **for** agent to N **do:** 11. Sample a random mini-batch of S samples  from D 12. Set 13. Update critic network:      1. Update actor network:      1. **end for** 2. Update target network parameters for each agent i:      1. **end for** 2. **end for** |

3.5 本章小结

本章描述了本文提出的MACTP模型的详细设计。MACTP模型的核心包括：多智能体任务建模和设计数值化的奖励反馈描述任务约束条件，以及使用多智能体Actor-Critic框架对该模型进行训练。这两个核心层面相互协作，从而使得MACTP模型可以有效应对智能无人集群的多目标动态规划问题，进而为多智能体协同完成相应任务提供了有效支撑。

# 第四章 基于分层注意力机制的多智能体强化学习算法

本章将对本文提出的基于分层注意力机制（Hierarchical Attention Mechanism，HAM）的MARL算法进行详细描述。在多目标动态规划任务中，当智能无人集群需要捕获的目标具有逃逸策略时，则该任务则属于混合任务。在混合任务中，无人集群中的智能体在学习策略时，由于智能体学习到的策略对接收的信息非常敏感，接收到信息不同可能导致智能体学习到不同的策略，因此需要同时考虑来自集群中的其他智能体信息和来自待捕获的目标智能体的信息，从而使智能体学习到正确的策略。在本章中，将智能无人集群中的智能体称为友方智能体，将待捕获的目标智能体称为敌方智能体。当环境中智能体数量增多时，在多智能体Actor-Critic框架中，Critic网络的输入空间维度快速增长，导致Q值的评估变得非常困难。为了解决Critic网络输入空间对智能体数量敏感的问题，本文提出了一个基于HAM的MARL算法（Actor Hierarchical Attention Critic，AHAC），该算法使用多智能体Actor-Critic框架，采用集中训练、分布执行的模式，利用RNN和HAM进行信息表示学习和信息压缩。本章将从算法整体框架开始，然后对算法的三个核心机制，即基于RNN的特征提取机制、基于HAM的信息压缩机制和多智能体Actor-Critic框架策略更新机制的设计和具体实现方法进行详细阐述。

4.1 AHAC算法框架

针对目标是具有逃逸策略的多智能体混合任务时，环境中智能体数量较多，Critic网络难以计算准确Q值的问题，我们提出了AHAC算法，其主要思想是利用基于HAM的集中式Critic网络，学习更加准确的Q值评估，使用表示。我们将混合任务中的智能无人集群按照智能体的任务目标分为多个群体，同一群体中的智能体仍处于合作关系中，不同群体之间为竞争关系。智能体既需要考虑友方智能体对其可能提供的帮助，也需要考虑敌方智能体对其带来的威胁。

AHAC算法框架的架构如图4.1所示。该算法使用多智能体Actor-Critic框架，主要由基于RNN的特征提取机制、基于HAM的信息压缩机制以及多智能体策略更新机制三个核心机制构成，AHAC算法框架内的算法流程如下。

首先，在图4.1的Environment部分，我们根据智能体的类型和智能体的任务目标，对智能体进行标记。其次，由于不同的智能体在混合任务中可能有不同的观察空间和动作空间，因此我们利用基于RNN的特征提取机制，将每个智能体的观察信息和动作信息投影到一个固定大小的特征空间中，保留重要信息。之后，为了解决集中式Critic网络的输入空间维度随智能体数量的增加而增长的问题，我们采用HAM对信息进行有侧重地压缩，基于HAM的信息压缩机制计算所有智能体的信息并将其压缩为一个固定长度的向量，该过程分为两个层级：第一层是智能体考虑与同一组中的其他智能体信息（包括其他智能体的观察值和动作值），根据同一组中其他智能体对其行为的影响，计算个体级重要信息；第二层是智能体考虑每个组的价值，根据不同组对其行为的影响，计算组级重要信息；通过HAM，智能体可以进一步判断威胁和帮助，并且固定智能体的Critic网络的输入空间维度，从而有效避免了Critic网络输出空间维度对智能体数量敏感的问题。最后，将压缩后的所有智能体的信息以及智能体自身的信息，输入集中式Critic网络计算Q值，Actor网络根据评估的Q值进行更新；并且，针对智能体策略泛化能力弱的问题，通过对智能体的动作进行实时采样的方式，实现智能体的策略更新，从而可以有效避免智能体无法根据当前的策略进行泛化的问题。智能体在环境中执行动作，进入后继状态，重复以上的训练过程。

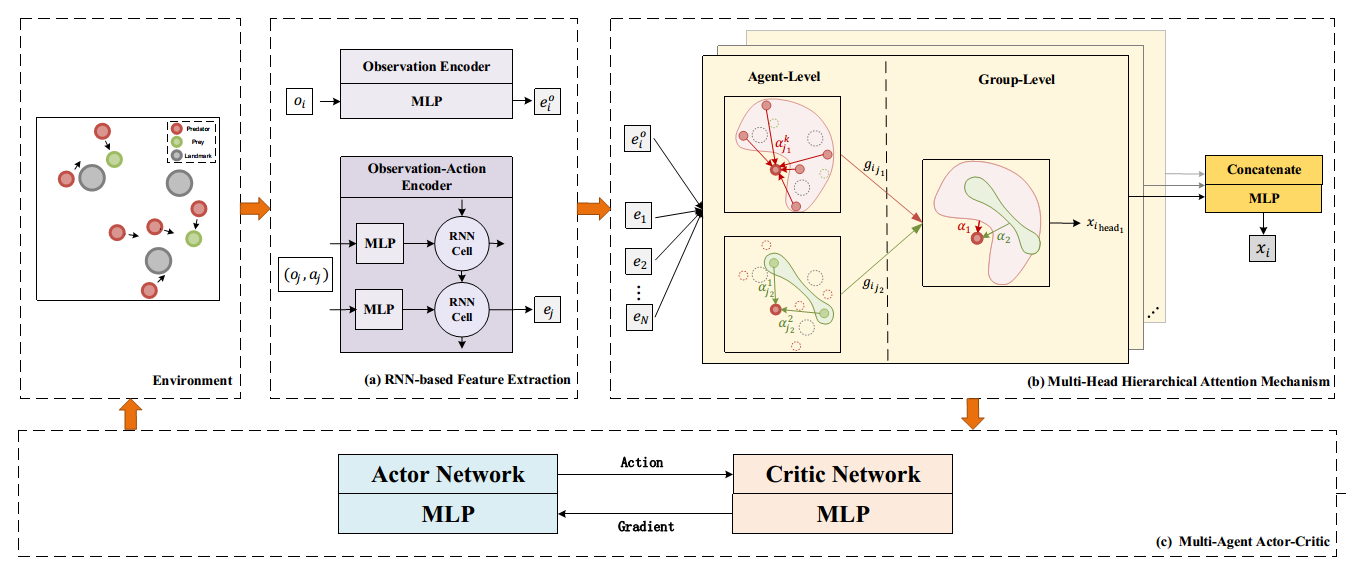


图4.1 AHAC算法整体框架设计

与MAAC算法[84]相比，AHAC算法具有如下两个方面的特点：一是使用基于RNN的特征提取机制对状态动作序列信息进行编码，将异构智能体的信息映射为相同维度的嵌入式；二是使用HAM对信息进行分组压缩处理，基于HAM的集中式Critic可以自适应地从多智能体提供的信息提取并保留重要信息，可以有效地帮助Actor网络调整策略。

4.2 基于RNN的特征提取机制

AHAC算法使用基于RNN的特征提取机制对接收的信息进行编码，将智能体的一系列原始数据（即观测信息和动作信息）转化为特定长度的、便于神经网络计算的特征向量，因此很好地可以解决异构多智能体问题中批评家接收到的信息长度不等的问题。在NLP中，RNN已经展现了处理序列信息的优越性[87]。在MDPs中，环境中的智能体接收当前的状态信息后，根据自身策略执行动作，进入下一个状态，并且环境会反馈给智能体奖励值[80]。由于智能体的决策训练过程遵循MDPs，因此智能体的动作与其当前的观察信息具有相关性。由于RNN隐藏层的神经元相互通信，状态值的输出信息存储在隐含层中。因此，与多层感知机（Multi-Layer Perceptron，MLP）相比，更适合使用RNN处理状态-动作序列信息[87]。

基于RNN的特征提取机制将从环境中获取的原始状态信息以及智能体的动作信息，计算状态-动作序列信息的嵌入式（Embedding）表达，联合其他智能体的贡献，作为集中式Critic网络的输入信息，输出智能体的估计Q值，如式（4.1）所示。

|  |  |
| --- | --- |
|  | （.1） |

其中，是一个两层的MLP；是智能体的状态-动作元组的嵌入式；是一个基于RNN的特征提取函数，由于RNN隐藏层间存在参数共享，所以能更好地处理状态-动作序列信息；和分别是智能体的观察值的嵌入式和动作值的嵌入式；是其他智能体对智能体的影响（或者是贡献），是多智能体的信息的加权结果，具体计算方式将在第4.3节中详细描述。

如图4.1中RNN-based Feature Extraction部分所示，在的计算过程中，我们利用两个一层的MLP，分别对智能体的观察值和动作值进行投影，得到智能体的观察值的嵌入式和动作值的嵌入式。将和进行拼接，作为特征提取函数的输入信息RNN计算单元的特性使得在计算动作值嵌入式时，状态值嵌入式也会影响最终的输出结果。因此，状态-动作元组的嵌入式不仅考虑了动作值和状态值各自的特点，而且考虑了状态值对动作值的潜在影响，因而能有效地提取状态-动作序列信息中的因果信息。

4.3 基于HAM的信息压缩机制

在AHAC算法中，设智能体的Q值函数为，集中式Critic网络接收所有智能体的观察值和动作值。我们将集群中除智能体之外的其他所有智能体标记为，并且利用对其进行索引。在中，我们将智能体的所有友方智能体利用进行索引，所有敌方智能体利用进行索引。

4.3.1 HAM具体组成

在MADDPG算法中，由于集中式Critic网络的输入空间的维度随智能体数量的增加而快速增长[61]，当环境中智能体的数量过多时，集中式Critic网络无法有效学习，从而输出的Q值无法对预期的奖励反馈进行准确拟合。因此，我们设计了基于HAM的集中式Critic网络，如图4.2所示，由预处理、个体级（Agent-Level）计算、组级（Group-Level）计算、信息整合以及集中式Critic网络五个部分构成，下面分别对这五个部分进行详细描述。

第一部分为预处理，主要作用是对来自其他智能体的信息进行映射处理，是对第4.2节的具体实现。针对智能体的集中式Critic网络，在HAM对其接收的信息加权压缩的过程中，智能体的观察值为，智能体的第个友方智能体的信息为，图4.2中的State Encoder为单层MLP，用于计算观察值的嵌入式；State-Action Encoder为基于RNN的特征提取网络，用于计算状态-动作序列信息的嵌入式。

第二部分为个体级（Agent-Level）注意力计算，主要作用是计算组内的智能体信息加权压缩后的组级信息向量。首先，计算智能体友方组内的智能体注意力权重的分布，Query、Key、Value分别为个体级的可学习的参数矩阵，Query、Key、Value具体含义可见第2.2节。将观察值嵌入式投影为个体级查询值，将状态-动作序列信息嵌入式分别投影为个体级键值和个体级价值，计算与之间的关联度并缩放后，进行归一化，与V进行点乘，得到友方组内所有智能体信息加权压缩后的信息向量。其次，智能体敌方组内智能体注意力权重的分布计算同理，第个敌方智能体的信息为，状态-动作序列信息的嵌入式为，敌方组内所有智能体信息加权压缩后的信息向量为。

第三部分为组级（Group-Level）注意力计算，主要作用是计算智能体友方和敌方的整体注意力权重的分布，从组与组之间的整体关系考虑所有智能体信息的重要性。设置Group\_Query、Group\_Key、Group\_Value分别为组级的可学习的参数矩阵，将观察值嵌入式投影为组级查询值，将组内信息向量分别投影为组级键值和组级价值，计算和之间的关联度并缩放，进行归一化，与进行点乘，得到第个注意力头上的所有智能体信息的加权压缩后的信息向量。

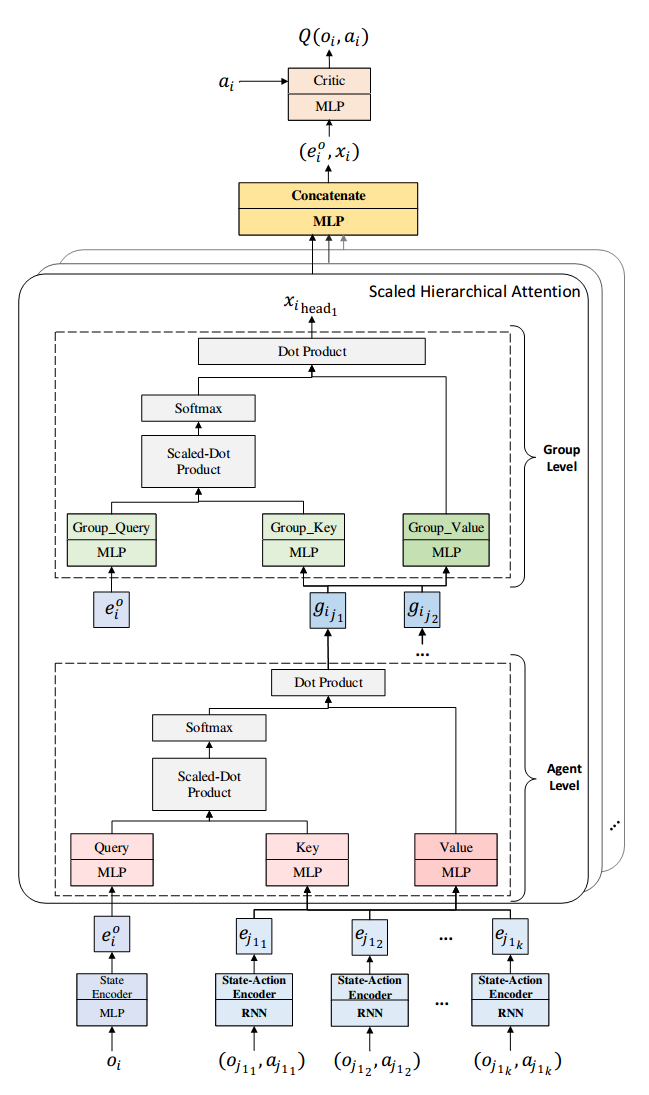


图4.2 利用分层注意力对信息进行加权压缩

第四部分为信息整合，主要作用是拼接所有注意力头的信息加权计算结果，得到所有注意力头上的其他所有智能体对智能体的综合贡献，联合智能体的观察值嵌入式，作为一个固定长度的信息向量，输入到集中式Critic网络中。

第五部分为集中式Critic网络，主要作用是根据输入的信息向量以及执行的动作值，计算得到Q值估计，具体更新过程见第4.4节。

4.3.2 HAM计算方法

HAM将所有智能体的信息加权成一个固定长度的向量，如式（4.2）所示。

|  |  |
| --- | --- |
|  | （.2） |

其中，NL代表非线性的激活函数，我们使用基于ReLU的LeakyReLU激活函数；代表可学习的参数矩阵，用于数据投影。在AHAC算法中，所有智能体的信息平行地在个独立的注意力头（即，特征子空间）上进行加权。是针对第个智能体的集中式Critic网络接收的信息，在第个注意力头上的计算结果，是智能体考虑到友方智能体和敌方智能体的影响后，每个智能体信息的价值的加权结果，如式（4.3）所示。

|  |  |
| --- | --- |
|  | （.3） |

其中，是一个可学习的参数共享的矩阵，用于对信息进行线性变换，得到组级价值；是表示智能体的友方（或敌方）的总体价值，即友方（或敌方）群组中的智能体的对智能体的加权影响，计算方法如式（4.4）所示。

|  |  |
| --- | --- |
|  | （.4） |

其中，代表友方（或敌方）中的第个智能体；是一个可学习的参数共享的矩阵，用于得到个体级价值；反映智能体对友方（或敌方）的关注程度，即组级的注意力权重；反映了智能体对群组中的每个智能体的信息关注程度，即个体级的注意力权重。在不同层级的注意力权重的计算中，（或）用于将原始信息转化为，（或）用于将原始信息转化为，计算并归一化Q与V的关联程度，根据这两个参数矩阵的维度对结果进行缩放，以防止梯度消失。

基于HAM的信息压缩机制的整体流程如表4.1所示。首先，根据输入信息，计算所有智能体的状态-动作序列的嵌入式和状态值的嵌入式，（第1-2行）。其次，对于智能体而言，根据其他智能体的类型和任务目标，将其他智能体分为智能体i的友方和敌方，即进行分组操作。基于HAM的信息压缩机制采用了多头注意力（Multi-head Attention），分别在多个头上进行个体级注意力和组级注意力计算，需要对多个头的结果进行拼接，随机初始化拼接参数矩阵（第4行）。在每个注意力头上，首先，初始化用于计算个体级查询值、键值、价值的可学习参数矩阵，用于计算组级查询值、键值、价值的可学习参数矩阵、、；其次，计算个体级注意力权重分布，即计算每个组内的其他智能体对智能体的影响，并将计算结果作为组级注意力计算的输入；然后，计算组级注意力权重分布，考虑友方和敌方对智能体的整体影响，计算结果即为在第个注意力头上，其他智能体对智能体行为的影响（第6-8行）。对所有注意力头的结果进行拼接并进行非线性映射，得到所有智能体对智能体的综合贡献（第10行）。最后，联合和，拼接为一个固定长度的向量，输入到集中式Critic网络中，计算得到智能体的状态-动作估计Q值（第11行）。

表4.1 基于HAM的信息压缩机制整体流程

|  |
| --- |
| 基于HAM的信息压缩机制的具体计算过程 |
| **输入：**所有个智能体的观察值信息和动作值信息  **输出：**智能体的状态-动作估计Q值   1. Calculate state-action embedding  for each agent ： 2. Calculate state embedding for each agent ： 3. According to the type of agent and task target, all agents are grouped into friend and enemy of agent 4. Initial learnable parameter matrix 5. **for** head=1 to head-num **do**： 6. Initialize Agent-Level learnable parameter matrix  and Group-Level learnable parameter matrix 7. In each group, calculate the influence of the state and action of agents in the group on agent as group-level input： 8. Consider the overall impact of each group, calculating the overall influence of friendly and enemy on agent ： 9. **end for** 10. Concatenate the results of all attention heads： 11. Calculate state-action Q value： |

值得注意的是，对于组级和个体级中的每个注意力头，用于提取、和的参数矩阵在所有智能体之间共享。即使在完全竞争的环境中，由于多智能体的Q值函数可以近似地看作是一个多任务回归问题，所以各智能体之间也可以共享参数。具有相同目标的智能体使用一组基于HAM的Critic网络。使用参数共享，可以使有不同的奖励反馈的智能体在有共同特征的环境中进行有效地学习。此外，有限的上下文长度往往成为限制模型性能的因素之一，因为输入信息的大小可能超出了固定长度上下文所能表达的范围。与其他基于注意力机制的CV和NLP方法对输入的上下文长度进行限制相比，由于在RL中每个智能体只提供观察值信息和行动值信息，AHAC算法对输入的时间或空间位置不做任何假设，所以AHAC算法中输入信息更为完整，具有一次计算即可获取全部输入信息中重要信息的优势。

AHAC算法采用了Multi-head Attention，每个注意力头都拥有两组独立的、所有智能体共享的参数矩阵，即个体级的和组级的，用于计算其他智能体对智能体的整体影响。AHAC算法对个体级信息进行加权，以获得组级的输入信息；在组级的权重计算中，需要将所有注意力头的计算结果拼接为一个固定长度的向量，作为集中式Critic网络的输入信息。Transformer将多个注意力头的计算结果进行拼接后再次投影，得到最终的上下文特征向量[78]。MAAC算法直接将多个注意力头的计算结果拼接作为其他智能体的加权后的信息[84]。而AHAC算法将多个注意力头的计算结果输入到一个非线性激活的前馈层，前馈层的输出结果作为其他智能体的加权后的信息，因此拟合重要信息的注意力权重分布更为准确。

4.4 多智能体Actor-Critic框架策略更新机制

多智能Actor-Critic框架中，智能体策略的更新流程如图4.3所示，采用集中训练、分布执行，并且我们在更新分布式Actor网络时，增加了优势函数的计算，用于评估智能体的动作对整体奖励反馈的影响。

4.4.1 多智能体Actor-Critic框架组成

AHAC算法采用的多智能体Actor-Critic框架中，主要包含了信息预处理、HAM、分布式Actor网络以及集中式Critic网络四个部分。其中，第一部分信息预处理的具体计算过程见第4.2节；第二部分HAM的详细设计及具体计算过程见第4.3节。

第三部分为分布式Actor网络，在多智能体Actor-Critic框架中，智能体的分布式Actor网络是独立的，输入信息为自身观察值。图4.3中代表智能体的策略，接收当前状态的观察值，选择执行动作值，智能体可以为其他智能体的Critic网络提供自身的状态-动作值，即交互信息。

第四部分为集中式Critic网络，主要作用是使用基于HAM的智能体的集中式Critic网络，获取所有智能体的观察值和动作值，HAM将所有智能体提供给智能体的信息压缩为信息向量，结合自身的信息，计算智能体的当前状态-动作的估计Q值。

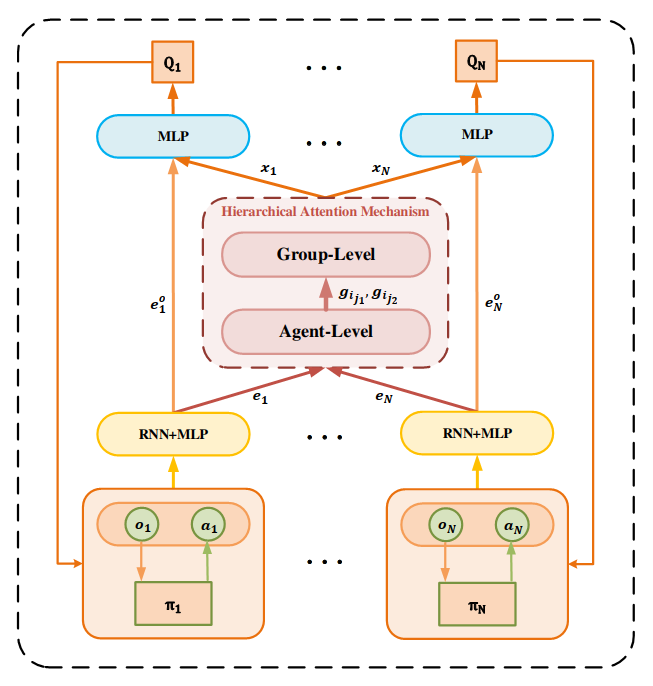


图4.3 基于HAM的多智能体Actor-Critic框架

值得注意的是，基于HAM的智能体的集中式Critic网络，可以获取所有智能体的观察值和动作，而智能体的Actor网络是独立的，只能获取自身的观察值。

4.4.2 多智能体Actor-Critic框架计算过程

在AHAC算法框架中，具有相同目标的智能体使用一组基于HAM的集中式Critic网络。与MADDPG算法中每个智能体的Critic网络独立更新相比[61]，由于参数共享，一组集中的基于HAM的Critic网络可以共同更新，从而使得如式（4.5）所示的联合损失函数最小化。

|  |  |
| --- | --- |
|  | （.5） |

其中，是智能体的动作-价值函数的估计，输入为所有智能体的观察值和动作值；和分别是智能体的目标Critic网络和目标Actor网络的参数；是温度参数，决定了最大熵和奖励反馈之间的平衡。

反事实基线利用了一个只边缘化指定智能体的行为的基线优势函数，解决多智能体的信用分配问题[54]。换言之，在固定其他所有智能体的动作的情况下，将智能体执行动作获取的Q值与智能体所有可能动作的平均Q值进行比较，可以有效判断预期奖励反馈增加的原因，是由于智能体当前的动作导致的，还是由于其他智能体的动作导致的。在连续动作空间中，可以通过抽样或者学习一个单独的权重矩阵，估计智能体的预期奖励反馈的期望。在离散动作空间的情况下，可以通过输出每个可执行的动作 的预期奖励反馈来计算智能体的基线，如式（4.6）所示。

|  |  |
| --- | --- |
|  | （.6） |

为了便于计算，我们将智能体的动作值从的输入信息中移除，并且在的输出信息中，加入了智能体的每个可执行的动作的Q值。因此，我们修改的输出维度为智能体可执行的动作的个数，增加了观察值编码器，使用替代之前描述的编码器。 为了进一步简化控制问题，我们对实验进行了改进，将智能体的动作空间设置为离散动作空间，利用梯度下降法对式（4.7）进行更新。

|  |  |
| --- | --- |
|  | （.7） |

在MADDPG算法中，Actor网络是通过从经验回放池中对智能体的行为进行采样来更新策略的[61]，这可能导致智能体无法根据当前的策略进行泛化的问题，而AHAC算法通过当前所有智能体的策略对智能体的动作进行采样，用于智能体的策略更新，从而避免了上述问题的发生。

4.5 AHAC算法

AHAC算法的核心是使用HAM对所有智能体的信息进行加权计算，嵌入式表达中保留重要信息，从而可以有效地解决集中式Critic网络的输入空间维度对环境中智能体数量敏感的问题。AHAC算法的完整伪代码如表4.2所示。

表4.2 AHAC算法：基于分层注意力机制的多智能体强化学习算法

|  |
| --- |
| 基于分层注意力机制的多智能体强化学习算法 |
| 1. Initialize E parallel environments with agents 2. Set replay buffer 4. **for** episode = 1 to **do:** 5. Reset all parallel environments 6. Receive initial observation 7. **for** to max-episode-length **do:** 8. Select actions  for each agent 9. Send actions  to all parallel environments and get new observation  and rewards 10. Store transition  for all parallel environments in D  13. **if** min-steps-per-update **then:** 14. **for** agent to **do:** 15. Sample a random mini-batch of samples  from 16. Preprocess the received information  and calculate the compressed information  for agent in parallel 17. Set 18. Update critic by minimizing the loss:      1. Set baselines: 2. Calculate advantage function: 3. Update policies by policy gradient:      1. **end for** 2. Update target network parameters for each agent :      1. **end if** 2. **end for** 3. **end for** |

在AHAC算法中，需要设置经验回放池，初始化个独立并行的环境（第1-3行）。在每一个训练周期中，首先对所有的独立并行的环境进行重置（第5-6行）。在每个环境中，智能体依靠Actor网络的决策，根据自身的观察值，从可执行的动作中进行选择并执行动作；执行动作后，智能体得到环境的奖励反馈值，并进入到下一个状态，得到新的观察值（第8-9行）。在每一个时间步，我们将所有智能体的四元组信息存储到经验回放池的同时，进行环境更新（第10-12行）。当经验回放池中存储了足够多的经验时，我们从中随机抽取一小批样本进行训练（第15行）。首先，我们通过HAM将每条样本中所有智能体的信息压缩成一个固定长度的向量（第16行），将其输入到集中式Critic网络中，计算策略的近似值，以及集中式动作-价值函数值，利用梯度下降法进行更新（第17-18行）。此外，AHAC算法还计算了一个多智能体基线，用于判断智能体执行动作是否会导致预期的奖励反馈的增加，或者奖励反馈的增加是否归因于其他智能体的联合动作（第19行）。同时，AHAC算法根据，计算优势函数，并且计算智能体的预期奖励反馈的梯度，使用Adam优化器，根据梯度更新其策略（第20-21行）。在每一个训练周期结束时，更新每个智能体的目标Critic网络和目标Actor网络的参数（第23行）。重复初始化环境和以上描述的训练过程，直到达到指定的最大训练周期数为止。

4.6 本章小结

本章详细描述了本文所提出的基于HAM的MARL算法——AHAC算法。AHAC算法的核心思想为：首先，使用基于RNN的特征提取机制，将原始信息转换为嵌入式；其次，使用基于HAM的信息压缩机制，将所有智能体的信息嵌入式加权计算，计算结果为一个固定长度的信息向量，并将其传入集中式Critic网络；最后，使用多智能体Actor-Critic框架策略更新机制，对所有的网络进行训练，保存的Actor网络模型即为智能体的策略，智能体根据训练后的策略，可以完成指定的多智能体任务。

# 第五章 原型系统实现与实验

本章针对上述两章描述的MACTP模型和AHAC算法进行了模块化实现，实现了一个基于MARL的多目标动态规划技术原型系统，在不同的实验场景中对系统性能进行了测试，并对MACTP模型和AHAC算法进行了对比实验，分析实验结果，验证了MACTP模型和AHAC算法的正确性和有效性。

5.1 原型系统架构与实现

5.1.1 环境配置

5.1.1.1 硬件环境配置

原型系统的详细硬件配置如表5.1所示。

表5.1 硬件环境详细配置

|  |  |
| --- | --- |
|  | 详细配置 |
| 硬件环境 | 处理器：Intel(R) Core(TM) i7-8700 CPU @ 3.20GHz |
| 内存：32.0GB |
| GPU：NVIDIA GeForce GTX 1050 Ti |

5.1.1.2 软件环境配置

原型系统的软件环境配置如表5.2所示。

表5.2 软件环境详细配置

|  |  |
| --- | --- |
|  | 详细配置 |
| 开发环境 | 操作系统：Windows 10 专业版 |
| 系统类型：64位操作系统，基于x64的处理器 |
| 开发环境：Python 3.7.3 |
| 仿真环境：Gym 0.9.4 |

开发环境所需的依赖项如表5.3所示。

表5.3 依赖项详细版本

|  |  |
| --- | --- |
| 名称 | 版本 |
| Gym | 0.9.4 |
| Torch | 1.2.0 |
| Numpy | 1.17.2 |
| TensorbordX | 1.8 |

5.1.2 原型系统设计

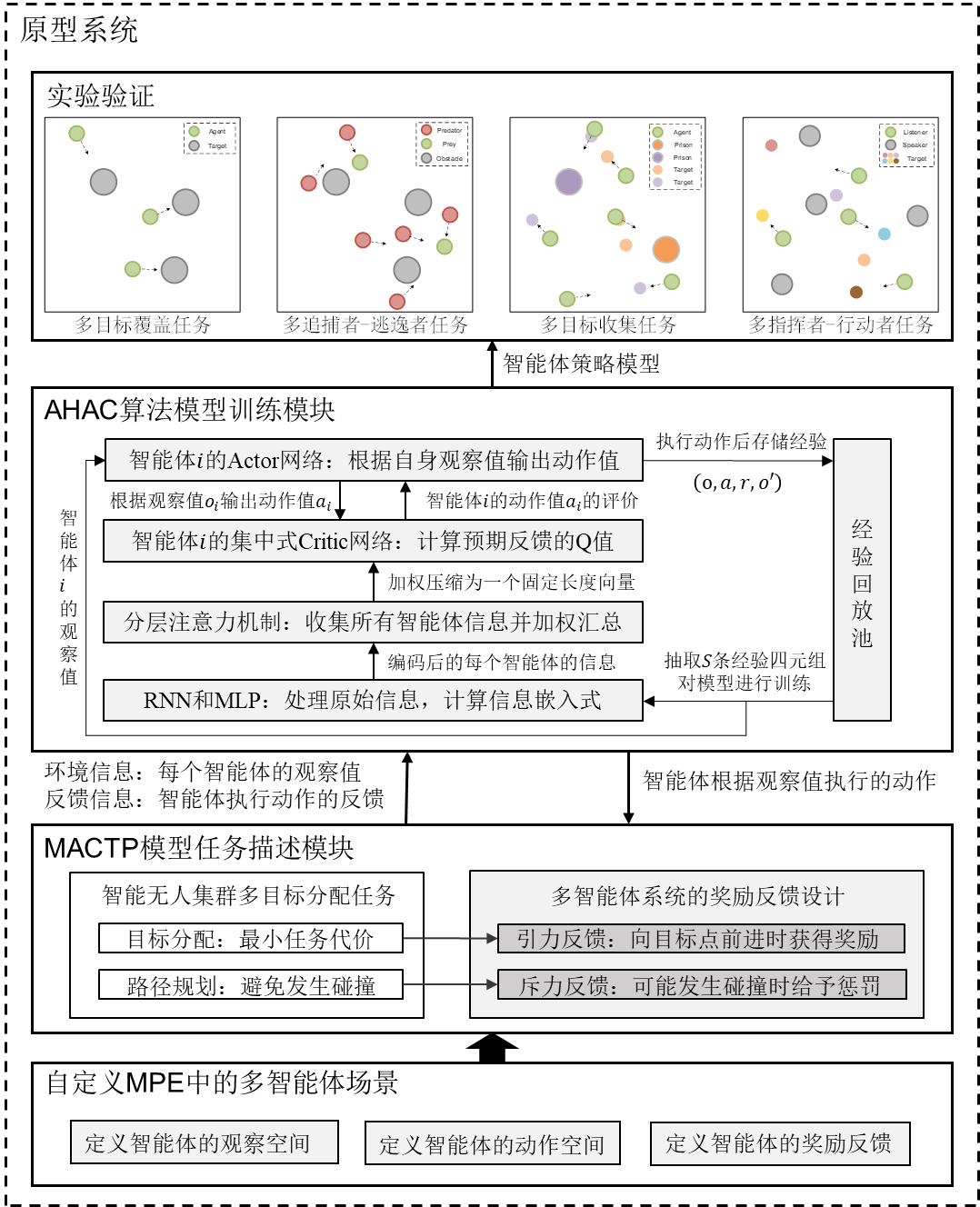


图5.1 原型系统总体架构设计

本文的原型系统架构如图5.1所示，主要包含了利用MACTP模型的任务描述模块和利用AHAC算法的模型训练模块，实现了基于MARL的多目标动态规划技术，使训练后的智能体可以适应未知的动态环境，提高了智能体策略的实时性和鲁棒性，并且具有一定的可扩展性。

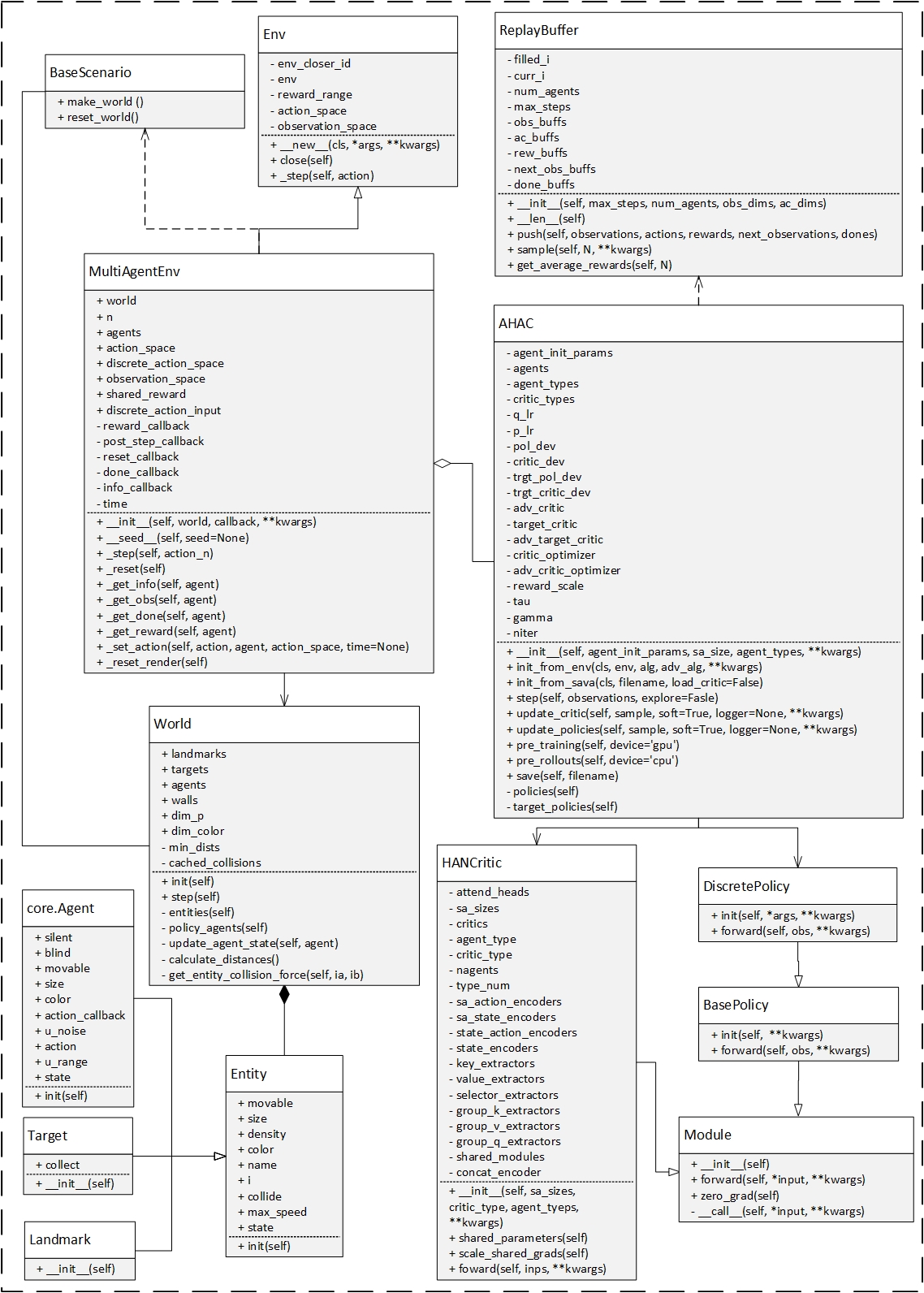


图5.2 原型系统UML图

该原型系统使用OpenAI开源的MPE作为实验环境，该实验环境允许开发者自定义实验场景，创建实体（智能体，目标点等），为实体分配功能，定义环智能体的观察空间、动作空间、奖励反馈，进行环境仿真。首先，利用MACTP模型描述多目标动态规划任务的约束条件，设计奖励反馈，解决了使用传统的多目标规划算法的智能体难以适应未知动态环境的问题，提升了智能体策略的鲁棒性和可扩展性；然后，利用AHAC算法对智能体的策略进行训练，解决了当目标是具有逃逸策略的智能体时，环境中智能体数量较多，Critic网络输入空间过大的问题，利用HAM对所有信息进行压缩，提升了智能体的学习效果，增强了MARL算法的可扩展性；最后，设计四个不同类型的多目标动态规划任务，对原型系统进行实验验证，证明了原型系统设计的正确性和有效性。

根据对MACTP模型的描述和AHAC算法的设计，使用Python语言对设计功能进行了实现。图5.2展示了原型系统中所包含的关键类，以及类与类之间的主要关系。其中，World类定义了环境中的实体类型以及实体功能，实体均继承Entity类，主要包含了描述智能体的Agent类、描述目标点的Target类以及描述环境中障碍物的Landmark类；Scenario类继承BaseScenario类，描述多智能体任务场景，实现MACTP模型；ReplayBuffer类实现了经验回放池的设计；AHAC类实例化HANCritic类和DiscretePolicy类，建立多智能体Actor-Critic框架，实现了AHAC算法。

5.1.2 关键算法实现

5.1.2.1 MACTP模型实现

MACTP模型主要实现了MAS中多智能体任务的构建和奖励反馈设计，通过构建MAS中的多智能体任务，设计奖励反馈描述多目标动态规划任务的约束条件，使用MARL算法对智能体的策略进行训练。我们使用Scenario类实现了第三章对MACTP模型的设计，Scenario类以及相关类的主要关系如图5.2所示。

Scenario类实现了MACTP模型，继承MPE的BaseScenario类。参数*num\_agents*、*num\_targets*和*num\_landmarks*分别定义了环境中智能体、目标点和障碍物的数量；参数*repulsion\_area*定义了斥力场的范围；参数*collect\_num*记录环境中被捕获的目标的数量。值得注意的是，在MACTP模型中，允许智能体和目标点的数目不一致。通过调用make\_world方法，设置环境中智能体、目标点和障碍物的数量以及属性（例如，尺寸，是否可移动等）；通过调用reset\_world方法对环境进行初始化，随机环境中实体的位置；通过调用is\_collision方法，判断智能体是否与其他智能体或者障碍物发生碰撞，调用is\_collision\_target方法，判断智能体是否已经到达（或者捕获）目标点；通过调用reward方法，评价智能体的动作，鼓励智能体尽快完成任务，获取智能体的奖励反馈；通过调用observation方法，获取智能体的观察值。

MACTP模型利用多智能体Actor-Critic框架，对Actor网络进行训练，即对智能体的策略进行训练。MACTP模型具体的实现及训练过程如表5.4所示。

表5.4 MACTP模型训练流程

|  |
| --- |
| MACTP模型训练流程 |
| 1. 初始化智能体、目标点和障碍物的数量以及属性，设置环境反馈给智能体的观察值； 2. 初始化经验回放池replay\_buffer； 3. 实例化环境，得到所有智能体的初始观察值序列； 4. 实例化基于Pytorch的多智能体Actor-Critic框架，每个智能体拥有一个Actor网络和一个Critic网络，初始化网络参数； 5. 在每一个训练周期内：    1. 将序列中的对应的各个智能体的观察值输入到各个智能体的Actor网络中，得到联合动作*action*；    2. 将联合动作*action*输入环境*env*的step函数中，所有智能体执行动作，与环境*env*进行交互，返回新的状态信息，包括下一状态的观察信息*next\_obs*序列和奖励反馈*reward*序列；    3. 将*obs*，*action*，*reward*，*next*保存为一个经验元组，存储到replay\_buffer中；    4. 判断是否需要更新网络参数且*replay\_buffer*中已经存储了一定数量的经验元组，如果满足条件，则从*replay\_buffer*中随机抽取*S*条经验更新网络；    5. 判断是否需要更新目标网络参数，如果是，则对目标网络参数进行软更新；    6. 将*next\_obs*赋值到*obs*；    7. 判断是否满足保存Actor网络模型参数的条件，如果满足，则保存模型参数；    8. 判断是否达到训练周期的最大步数*max\_episode\_len*或者完成目标规划任务，如果满足，则终止本训练周期，重置*env*，否则进入步骤5.9；    9. 返回步骤5.1； 6. 判断是否达到最大的训练周期数*n\_episodes*，如果是，则保存模型参数，结束训练。 |

5.1.2.2 AHAC算法实现

我们使用HANCritic类和AHAC类实现了第四章对AHAC算法的设计，HANCritic类和AHAC类以及相关类的主要关系如图5.2所示。

HANCritic类实现了基于HAM的集中式Critic网络的设计，在实例化HANCritic类时，参数*sa\_sizes*定义了每个智能体的观察值维度和动作值维度，参数*critic\_type*定义了当前集中式Critic网络是否采用HAM，参数*attend\_heads*定义了注意力头的数量，参数*agent\_type*定义了环境中智能体的类型，参数*type\_num*定义了环境中智能体的种类数量，*hidden\_dim*定义了网络隐藏层的维度；为每个智能体建立一个四层神经网络的Critic网络；为每个智能体建立一个三层神经网络的观察值编码器；为每个智能体建立一个基于RNN的状态-动作编码器，包含输入层、一层隐藏层、两层RNN计算单元和输出层；为每一个注意力头，建立个体级的可学习的查询值参数矩阵、键值参数矩阵和价值参数矩阵，组级的的可学习的查询值参数矩阵、键值参数矩阵和价值参数矩阵；为Multi-head Attention建立一个三层神经网络的结果拼接器。调用shared\_parameters方法和scale\_shared\_grads方法，实现神经网络的参数共享。定义forward方法，显示定义了上述网络的计算图，具体计算过程如表5.5所示。

表5.5 HANCritic类的计算图描述

|  |
| --- |
| HANCritic类张量运算 |
| **输入：**所有智能体的观察值和动作值  **输出：**智能体可获得最大Q值的动作及对应的Q值，智能体所有可执行动作的Q值   1. 智能体i的观察值输入到观察值编码器，输出智能体i的观察值的嵌入式s\_encodings； 2. 智能体的观察值和动作值输入到状态-动作编码器，输出智能体的状态-动作的嵌入式*sa\_encodings*； 3. 在注意力头k中，将s\_encodings输入到个体级查询值参数矩阵中，输出个体级查询值all\_head\_selectors，将sa\_encodings分别输入到个体级键值参数矩阵和个体级价值参数矩阵中，输出个体级键值all\_head\_keys和个体级价值all\_head\_values； 4. 使用张量乘法计算all\_head\_selectors和all\_head\_key关联程度并根据矩阵维度进行缩放，得到scaled\_attend\_logits； 5. 对scaled\_attend\_logits进行归一化，得到组内的个体级注意力权重attend\_weights； 6. 使用矩阵乘法计算attend\_weights和all\_head\_values，得到加权压缩后的组内所有智能体信息other\_values； 7. 在注意力头中，所有组的张量结果other\_values进行拼接，得到张量group\_v\_encoding，将其分别输入到组级键值参数矩阵和组级价值参数矩阵中，输出组级键值group\_key和组级价值group\_value；将all\_head\_selectors输入到组级查询值参数矩阵中，输出组级查询值group\_query； 8. 使用张量乘法计算group\_query和group\_key关联程度并根据矩阵维度进行缩放，得到group\_attend\_logits； 9. 对group\_attend\_logits进行归一化，得到组间的组级注意力权重group\_attend\_weights； 10. 使用矩阵乘法计算group\_attend\_weights和group\_value，得到加权压缩后的所有智能体信息all\_other\_values； 11. 将所有注意力头的输出张量进行拼接，输入到结果拼接器，输出*c\_other\_value*； 12. 将张量*c\_other\_value*和张量*s\_encoding*进行拼接，输入到Critic网络，输出智能体所有可执行动作的Q值； 13. 计算智能体可获得最大Q值的动作及对应的Q值； 14. 返回计算结果。 |

AHAC类实现了AHAC算法的设计，包含了实例化HANCritic类以及更新Actor网络和Critic网络。参数*critic\_types*定义了包含的集中式Critic网络的组数，一组集中式Critic网络一起更新；参数*nagents*定义了智能体的数量；参数*p\_hidden\_dim*和*q\_hidden\_dim*分别定义了Actor网络和Critic网络的隐藏层的神经元数量；参数*gamma*定义了对平衡未来预期奖励反馈的衰减因子；参数*tau*定义了软更新目标Actor网络和目标Critic网络时的原有参数保留程度。调用init\_from\_env方法，从多智能体环境中实例化AHAC类；调用policies方法，获取所有智能体的Actor网络；调用target\_policies方法，获取所有智能体的目标Actor网络；调用step方法，输入所有智能体的观察值，输出所有智能体的执行动作，智能体在环境中执行动作；调用update\_critics方法，输入从经验回放池随机采样的样本，对所有智能体的Critic网络更新；调用update\_policies方法，输入从经验回放池随机采样的样本，对所有智能体的Actor网络更新；调用update\_all\_targets方法，对所有智能体的目标Actor网络和目标Critic网络进行软更新；调用save方法，保存网络参数，如果满足设定的保存模型条件，则进行模型保存操作。

AHAC算法训练流程如表5.6所示。

表5.6 AHAC算法训练流程

|  |
| --- |
| AHAC算法训练流程 |
| 1. 随机生成并行环境的初始，实例化训练环境*env*； 2. 实例化AHAC类，创建*model*； 3. 如果不是对进行训练，则从指定文件路径读取已保存的模型参数； 4. 实例化经验回放池*replay\_buffer*； 5. 在一个训练周期内：    1. 重置训练环境*env*；    2. 获取每个智能体的策略*policy*；    3. 获取所有智能体的观察值*obs*序列    4. 将序列中的对应的各个智能体的观察值输入到各个智能体的Actor网络中，得到联合动作*action*；    5. 将联合动作*action*输入*env*的step函数中，所有智能体执行动作，与*env*进行交互，返回新的状态信息，包括下一状态的观察信息*next\_obs*序列和奖励反馈*reward*序列，以及其他信息*infos*；    6. 将*obs*，*action*，*reward*，*next*，*infos*保存为一个经验元组，存储到*replay\_buffer*中；    7. 判断是否对*model*进行训练，即对模型测试，则返回步骤5；    8. 如果*replay\_buffer*中存储足够多的经验元组，更新网络参数的更新步数*num\_updates*内：       1. 从*replay\_buffer*中随机抽取*S*条经验，更新Critic网络；       2. 从*replay\_buffer*中随机抽取*S*条经验，更新Actor网络；       3. 如果达到设定的延迟更新步数*updtae\_nums*，对目标网络参数进行软更新；    9. 判断是否达到训练周期的最大步数*max\_episode\_len*，如果满足，则结束该训练周期，进入步骤，否则，返回步骤5.1； 6. 计算本训练周期内的平均奖励反馈*ep\_rews*； 7. 保存本次的训练数据到日志文件*logger*中，并保存本次训练周期得到的训练模型； 8. 判断是否达到最大的训练周期数*n\_episodes*，如果是，则保存模型参数，结束训练。 |

5.2 算法实验与结果分析

5.2.1 实验基线

5.2.1.1 MACTP的基线

我们将MACTP模型与STAPP方法[45]进行了对比实验。STAPP方法虽然使用MARL算法解决智能无人集群的多目标动态规划问题，但是与MACTP模型相比，STAPP方法具有更强的约束条件，要求环境中智能体和目标点的数量必须一致，只能解决对称多目标动态规划问题。

同时，我们考虑了MACTP模型的简化版本，使用稀疏奖励反馈描述多目标动态规划任务的约束条件，即仅当捕获到目标时获得一个正向奖励反馈，与其他智能体或者障碍物发生碰撞时，则获得一个负向惩罚反馈，不存在过渡奖励反馈，从而验证MACTP模型中过渡奖励反馈设计的有效性。在实验中，使用MACTP-Spar表示该简化版本。

5.2.1.2 AHAC算法的基线

我们将AHAC算法与MADDPG算法[61]、MAAC算法[84]进行了对比实验。MADDPG算法使用Gumbel SoftMax重参数化[86]，使其训练的智能体的策略可以用于离散动作空间。同时，在实验过程中发现，MAAC算法在混合任务上的效果表现较差。为了提高在混合任务下MAAC算法训练的智能体的学习效果，MAAC算法在更新过程中，所有智能体不再仅使用一组注意力机制，而是参考AHAC的设计，具有相同目标的智能体共享一组注意力机制，并将该组内的Critic网络一起更新。为了方便描述，我们将在混合任务中具有更好训练效果的改进后的MAAC算法称为MAAC-Adv。

同时，我们考虑了AHAC算法的简化版本，分别是仅使用HAM的AHAC-HAN和仅使用RNN和单层注意力机制的AHAC-RNN。通过对比AHAC算法和AHAC-HAN的实验效果，可以验证RNN是否可以学习包含在状态-动作序列信息中的隐藏信息。对混合任务中，对比AHAC算法和AHAC-RNN实验效果，可以验证分开计算友方和敌方影响的必要性。

AHAC算法的所有基线的主要信息如表5.7所示。

表5.7 不同的多智能体强化学习算法的对比

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 基线 | 信息聚合方式 | 信息编码  方式 | 混合任务中，Critic网络的组数 | 可解决的多智能体  任务类型 |
| AHAC | HAN | RNN+MLP | 2 | All |
| AHAC-HAN | HAN | MLP | 2 | All |
| AHAC-RNN | Attention | RNN+MLP | 2 | All |
| MAAC | Attention | MLP | 1 | Full cooperative tasks |
| MAAC-Adv | Attention | MLP | 2 | All |
| MADDPG | (Observation, Action) | 无 | N | All |

5.2.1.3 参数设置

实验过程中，所有神经网络使用的超参数设置如表5.8所示。

表5.8 神经网络超参数设置

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数含义 | 默认值 |
| n\_rollout\_threads | 并行训练环境数量 | 6 |
| buffer\_length | 经验回放池容量 | 1000000 |
| n\_episodes | 最大训练周期数量 | 50000 |
| steps\_per\_update | 每个Episode中最大步数 | 100 |
| num\_updates | 每轮更新的间隔步数 | 4 |
| batch\_size | 每次抽取样本数量 | 1024 |
| gamma | 未来预期收益衰减因子 | 0.99 |
| lr | 神经网络学习率 | 0.001 |
| hidden\_dim | 隐藏层神经元数量 | 128 |

5.2.2 实验环境

|  |  |
| --- | --- |
| 图5.3 多目标覆盖任务示意图 | 图5.4 多追捕者-逃逸者任务示意图 |
| 图5.5 多目标收集任务示意图 | 图5.6 多指挥者-行动者任务示意图 |

我们在MPE中设计了4种类型的多目标动态规划任务，对我们提出的MACTP模型和AHAC算法进行测试，设计的实验任务包括了多智能体完全合作任务和多智能体混合任务。我们进一步简化了控制问题，并使用离散动作空间。由于MPE使用了一个基本的物理引擎，考虑了智能体的动量，智能体在做出动作后，可能不会立即准确地向指定的方向移动。

多目标覆盖任务场景如图5.3所示，在该场景中，个智能体通过协作，尽可能到达预设的个目标点。虽然所有智能体仅进行物理动作，但是可以观察到其自身与其他智能体和目标点的相对位置。智能体的奖励反馈与每个目标到任意一个智能体的距离有关，所以要求智能体必须覆盖每一个目标，才能获得最大奖励。并且，当智能体与其他智能体碰撞时，会获得惩罚。我们尝试让智能体学到覆盖所有目标，并且在移动过程中避免碰。该任务场景为最简单的多智能体完全合作任务场景。

多追捕者-逃逸者任务场景如图5.4所示，在该场景中，存在个速度较慢并且具有合作关系的追捕者，个速度较快的逃逸者（），并且存在个随机生成的障碍物地标（）。每个智能体都可以观察到自身与其他智能体和障碍物的相对距离。在每个时间步中，追捕者若碰撞到逃逸者，则追捕者获得正向奖励反馈，逃逸者获得惩罚。同时，环境存在边界，当逃逸者碰撞到边界时，也会得到额外惩罚。该场景属于混合场景，智能体间不仅存在合作，也存在竞争。

多目标收集任务场景如图5.5所示，在该场景中，存在个目标点和8个合作的智能体，其中6个是目标寻找者，2个是目标收集者。每个目标收集者对应不同颜色（即不同类型）的目标。目标寻找者可以收集任何颜色的目标，并把收集到的目标放在相应颜色的目标收集者处。每个目标收集者的职责是从目标寻找者处收集尽可能多的目标。智能体可以观察其他智能体的相对位置。目标寻找者成功收集目标点可获得整体奖励，目标收集者收集目标后，所有智能体都可以获得奖励。智能体间若发生碰撞，则受到额外惩罚。该任务场景属于包含了整体和个人奖励的完全合作任务场景。

多指挥者-行动者任务场景如图5.6所示，在该场景中，存在个目标点（），8个合作的智能体，其中4个是行动者，4个指挥者。在环境初始化过程中，指挥者和行动者随机一一匹配。在该环境中，行动者无法在视野中直接看到目标，必须依靠和指挥者的通信来到达目标。指挥者可以定位行动者和行动者需要到达的目标，并向对应的行动者发送离散的通信信息。指挥者和行动者的奖励反馈依赖于行动者到目标的距离，距离越近，奖励越高。值得注意的是，在这个场景中，指挥者只有通信动作，行动者只有物理动作。在我们的环境设置中，通信属于行动者的观察空间的一部分，而不是最后学习得到的模型的一部分。该任务场景属于存在异构智能体和特定奖励反馈的完全合作任务场景。

5.2.3 MACTP模型实验与分析

我们在如图5.3所示的多目标覆盖任务中，对MACTP模型以及MACTP模型的基线进行了测试。

5.2.3.1 有效性

为了证明MACTP模型解决非对称多目标动态规划任务时（即，智能体的数量和目标点的数量不对等）的有效性，在多目标覆盖任务场景中，设定智能体的数量为3，障碍物的数量为4，在具有不同目标点数量的情况下，分别使用MACTP模型以及MACTP模型的基线，对智能体进行训练，记录智能体在每个训练周期覆盖到的目标点的数量、发生碰撞的次数以及完成任务所需的步数。

|  |  |
| --- | --- |
| 图5.7 多目标覆盖任务中存在3个目标点时的覆盖情况 | 图5.8 多目标覆盖任务中存在4个目标点时的覆盖情况 |
| 图5.9 多目标覆盖任务中存在5个目标点时的覆盖情况 | |

如图5.7至图5.9所示，记录了在多目标覆盖任务中，环境中存在不同数量的目标点的情况下，智能体在每个训练周期覆盖到的目标点的个数。由于在该任务中，智能体的目标是覆盖尽可能多的目标点，所以该指标可以直接反馈智能体的训练效果。红色曲线为MACTP模型的表现，相比于基线，当目标点数量变多时，训练效果具有明显的提升。由于环境是随机初始化的，智能体和目标点在环境中的分布可能较为分散，且环境中存在障碍物，一个训练周期的最大步长规定为100，可能存在智能体在一个训练周期内未能覆盖所有目标点的情况，所以我们不仅考虑平均覆盖目标点曲线的趋势，同时也考虑了曲线的峰值，即一个训练周期内，智能体最多覆盖目标点的个数。如图5.7所示，当环境中智能体和目标点的数量均为3时，此时该任务场景属于对称多目标动态规划任务，利用STAPP方法训练的智能体可以覆盖一到两个目标点。在该对称多目标覆盖任务场景下，MACTP模型也取得了较好的效果，在每个训练周期内基本可以覆盖到两个目标点。值得注意的是，虽然MACTP-Spar也可以覆盖一到两个目标点，但是由图5.7中可以明显看到MACTP-Spar的训练效果并不稳定，这是由于MACTP-Spar采用稀疏奖励，智能体仅探索到关键状态，即覆盖目标点或者与发生碰撞时，才会获得正向奖励或者负向惩罚，缺乏过度奖励反馈设计的激励，所以智能体的策略难以收敛。

当任务场景属于非对称多目标动态规划任务，即环境中智能体数量不变，仍设定为3个，目标点数量变多时，如图5.8和图5.9所示，环境中分别存在4个目标点和5个目标点时，STAPP方法仍最多只可以覆盖两个目标点。经过分析，在STAPP方法中，由于奖励反馈设计中，缺乏对已经覆盖的目标点的判断，所以智能体会停留在已经覆盖的目标点周围，重复获得奖励反馈，不再探索环境中的未知状态，从而无法覆盖新的目标点。同理，在MACTP-Spar模型的奖励反馈设计中，覆盖目标点则获得奖励，缺乏对目标点覆盖状态的判断，所以MACTP-Spar模型在非对称多目标动态规划任务中，缺乏对未知状态空间的探索，表现基本和STAPP方法持平。如图5.8和图5.9所示，MACTP模型相较于基线，覆盖的目标个数明显提升，证明MACTP模型在训练过程中可以做到完全覆盖所有目标点，顺利完成非对称多目标覆盖任务。

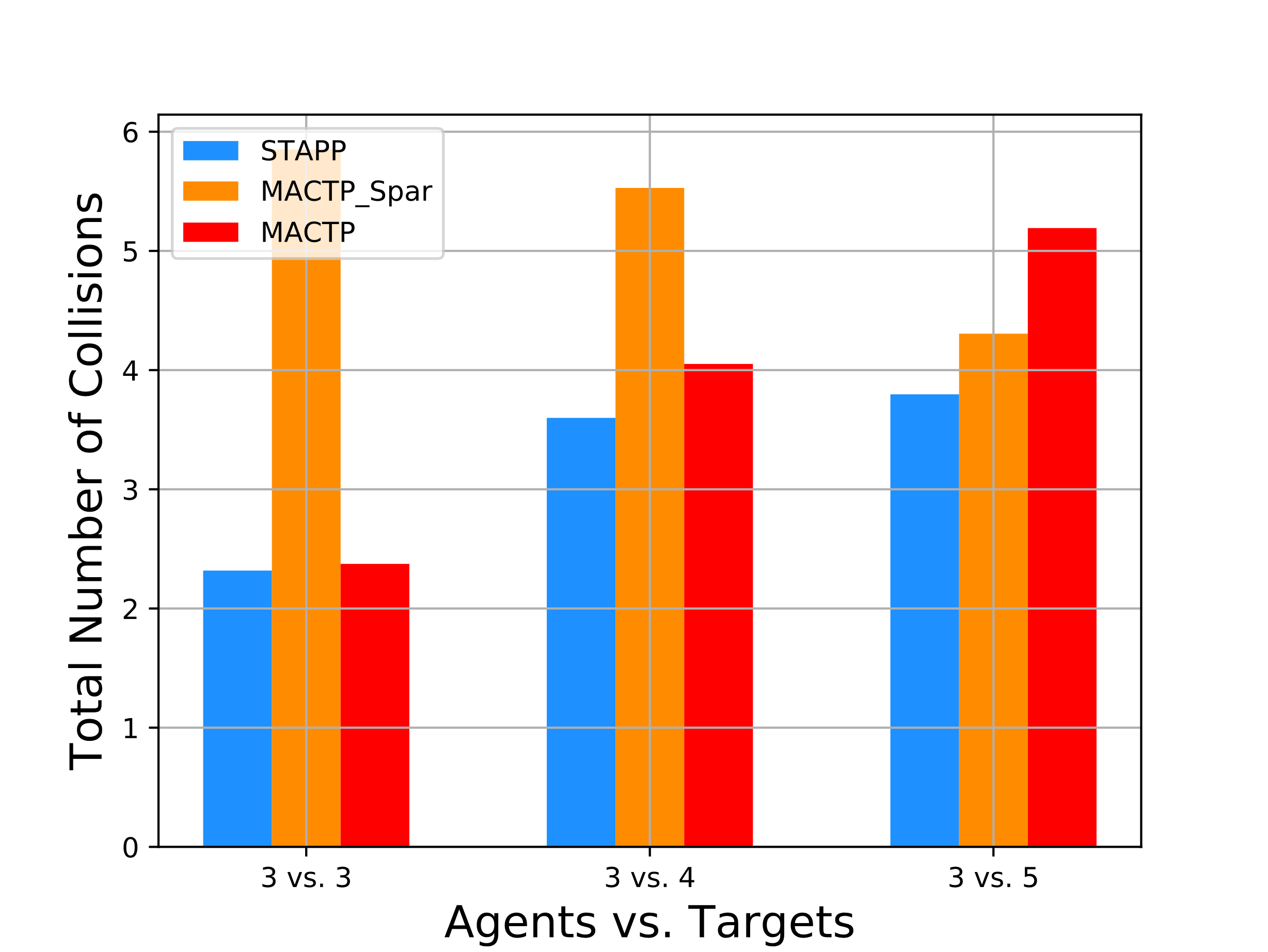


图5.10 存在不同数量目标点时，智能体与其他智能体或者障碍物发生碰撞的总次数

如图5.10所示，当环境中存在固定数量的智能体和障碍物，在设定不同数量的目标点的情况下，我们记录了智能体在完成任务的过程中，碰撞到其他智能体或者障碍物的总次数。当环境中存在3个智能体和3个目标点时，STAPP方法和MACTP模型的碰撞次数基本一致，而MACTP-Spar模型的碰撞次数比较高。由于MACTP-Spar缺乏过渡奖励反馈，即不存在对碰撞的预警，只有碰撞后智能体才会受到惩罚。由于智能体的任务目标是覆盖目标点，所以智能体为了完成任务，获得更高的奖励反馈，可能无视碰撞带来的惩罚，所以MACTP-Spar模型具有较高的碰撞总次数。随着目标点数量的增多，MACTP模型的碰撞次数逐渐增多，而STAPP方法的碰撞次数变化较小，这主要是由于STAPP方法缺乏鼓励智能体探索未知的环境状态，当智能体覆盖到目标点时，则一直徘徊在该目标点附近，重复获得奖励，所以可以避免发生新的碰撞，但是未能达到覆盖所有目标点的任务要求。同理，MACTP-Spar模型使用稀疏奖励，智能体探索未知空间并不能及时获得奖励反馈，智能体在覆盖到目标点后，则不再继续探索，所以随目标点的增多，MACTP-Spar模型的智能体的碰撞次数基本不变。而MACTP模型中的奖励反馈设计包含了对目标点状态的判断，当智能体覆盖一个目标点后，修改目标点的覆盖状态，智能体将不能再从该目标点获得奖励反馈，鼓励智能体探索未知的环境，覆盖更多目标点，完成非对称多目标动态规划任务。但是在探索过程中，仍可能与其他智能体或者障碍物碰撞，所以智能体的碰撞次数多于STAPP方法和MACTP-Spar模型训练的智能体。

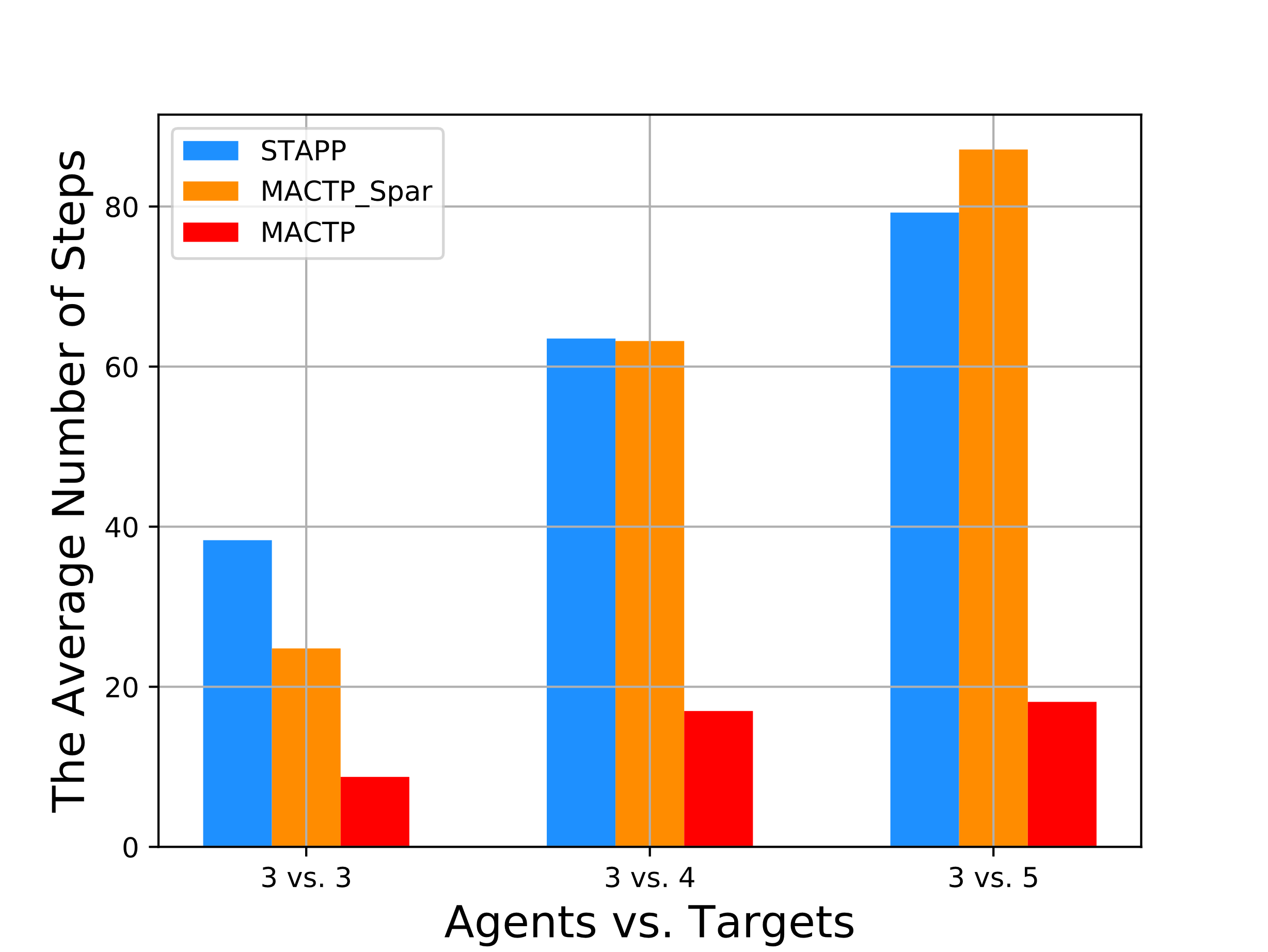


图5.11 不同数量目标点时，智能体覆盖所有目标点需要的步数

在多目标动态规划任务中，一个重要的约束条件是最小化运动代价。在多目标覆盖任务中，智能体的运动代价为覆盖所有目标点所需的步数。为了验证MACTP模型的正确性，即可以满足运动代价最小化的约束条件，我们记录了当智能体数量固定为3，不同数量目标点的多目标覆盖任务中，智能体在一个训练周期内完成任务所需的平均步数，即智能体需要覆盖所有目标点，智能体完成任务所需的平均步数如图5.11所示。随着目标点数量的增多，STAPP方法和MACTP-Spar模型训练的智能体完成任务的步数明显增多，而MACTP模型训练的智能体完成任务的步数始终控制在20步以内，证明MACTP模型通过奖励反馈设计，降低了模型对目标点数量的敏感性，提高了智能体的鲁棒性和可扩展性。

同时，由图5.11的实验结果可以看出，当任务场景中的智能体数量不变，存在的目标点数量分别为3个、4个和5个时，MACTP模型训练的智能体在每个任务场景中完成任务所需的平均步数均小于SATPP模型和MACTP-Spar模型。该实验结果验证了基于人工势场法的奖励反馈设计的有效性，智能体不仅在覆盖目标点时的关键状态可以获得奖励反馈，在非关键状态时，通过智能体与目标点的距离，为智能体提供引力反馈，可以鼓励智能体快速向目标点前进，避免智能体进行无意义的环境探索，降低智能体完成任务所需的运动代价。

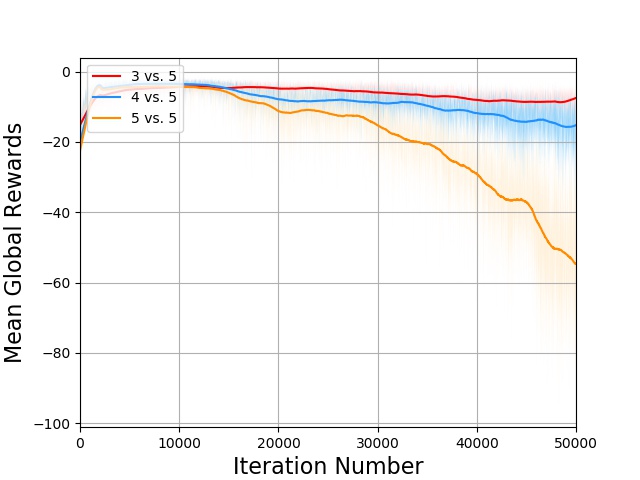


图5.12 不同数量智能体时，智能体获得的平均全局奖励反馈

同时，我们在该任务场景中，保持障碍物数量不变，设定目标点的数量为5，增加智能体的数量，测试了模型在智能体数量增多时的可扩展性。如图5.12所示，随着智能体数目的增加，智能体获取的整体奖励反馈变的越来越小，收敛也变的越来越困难，说明智能体学习到有效的策略的难度越来越大。经过分析，主要原因在于集中式Critic网络的输入空间随着智能体数量的增多而快速增长，增加了Q值计算的难度，难以给Actor网络一个准确的反馈。因此，如何在环境中存在更多智能体的情况下，提高集中式Critic网络的反馈准确程度和训练效率，是提高MACTP模型训练效果的工作方向之一。

5.2.3.2 斥力场范围设定

在MACTP模型中，我们参考人工势场法的思想，设计并实现了了斥力场。为了验证斥力场的有效性并探索斥力场范围的设定，我们在多目标覆盖任务中，设置智能体的数量为3，障碍物的数量为2，目标点数量为5，斥力场的范围分别为RF=0.0，0.1，0.2和0.4时，对MACTP模型的性能进行测试，记录在完成任务的过程中，智能体碰撞到其他智能体或者障碍物的次数，如图5.13所示；智能体覆盖所有目标点所需的平均步数，如图5.14所示。其中，当斥力场的范围设定为0.0时，实际是未设置斥力场，则智能体不会受到斥力反馈。

|  |  |
| --- | --- |
| 图5.13 不同RF范围时，发生碰撞的次数 | 图5.14 不同RF范围时，完成任务的步数 |

由图5.13的实验结果对比可以看出，斥力场的设置可以降低智能体与其他智能体或者障碍物发生碰撞的概率，增大斥力场范围，发生碰撞的次数明显减小。在实际应用中，智能无人集群不仅需要考虑避免碰撞，降低损毁率，并且需要考虑续航问题，尽量在最小的代价内完成任务，所以需要考虑智能体完成任务所需的步数。如5.15所示，随着斥力场范围的增大，智能体完成任务的平均步数也逐渐增大。因为在斥力场内，智能体不仅受到目标点的引力反馈，也受到其他智能体或者障碍物的斥力反馈，智能体无法直接向目标前进，从而影响完成目标的效率。所以，在综合考虑碰撞次数和完成任务步数的基础上，我们将斥力场的范围设定为RF=0.2，在较低的碰撞次数的情况下，以最小的运动代价完成任务。

5.2.4 AHAC算法实验与分析

我们在4种类型的多目标动态规划任务中，如图5.3至图5.6所示，对AHAC算法以及AHAC算法的基线进行了测试，记录了每个训练周期的平均奖励，如图5.16至图5.19所示。实验结果证明AHAC算法的训练效果具有明显的提升，尤其是在混合任务中，AHAC算法表现更为出色。我们从以下五个方面进行分析：HAM的影响、可扩展性、多头注意力结果的拼接方式、注意力头的数量以及注意力权重分布。下面分别进行详细介绍。

**5.2.4.1 HAM的影响**

如图5.17所示，我们提出的方法在多追捕者-逃逸者（6 vs. 2）任务中具有明显的性能优势。在该任务中，智能体数量多，交互信息复杂，导致MADDPG算法效果不佳。MAAC算法在这项任务中亦表现不佳，主要原因在于：MAAC算法因为计算所有智能体的Critic网络的累计误差并对所有Critic网络进行集中更新，Critic网络不能对预期收益做出合理的预测。MAAC-Adv保证只对具有相同目标的智能体的Critic网络进行集中更新，从而获得更好的结果。AHAC算法利用HAM计算来自朋友和敌人的信息的价值并对信息进行压缩，因此AHAC算法比MAAC算法、 MAAC-Adv的性能更好。AHAC算法首先通过个体级注意力机制，计算所有友方智能体的信息的重要性分布，计算获得友方智能体信息的价值。通过同样的计算，获得敌方智能体信息的价值；在组级注意力方面，AHAC算法计算友方信息和敌方信息的整体价值，并进行加权计算，进而获得所有智能体的有效信息。相对于AHAC算法，AHAC-RNN则直接对所有智能体的信息进行加权，得到所有智能体的有效信息。由于AHAC算法对友方和敌方整体信息的计算，所以在多追捕者-逃逸者（6 vs. 2）任务中比AHAC-RNN性能更好。从图5.17可以看出，AHAC-HAN学习的策略不稳定，最终效果略差于AHAC算法。显然，使用RNN对状态-动作信息编码，可以更好地对原始信息进行预处理。

|  |  |
| --- | --- |
| 图5.16 多目标覆盖（3 vs. 3）任务中，智能体获取的平均收益 | 图5.17 多追捕者-逃逸者（6 vs. 2）任务中，追捕者获取的平均收益 |
| 图5.18 多目标收集任务中，智能体获取的平均收益 | 图5.19 多指挥者-行动者任务中，智能体获取的平均收益 |

由于多目标覆盖任务、多目标收集任务和多指挥者-行动者任务属于多智能体完全合作任务，环境中不存在敌方智能体，所以AHAC算法不需计算组级注意力，只使用RNN和单层的多头注意力机制。并且，MAAC算法可以应用于多智能体完全合作任务，不需使用MAAC-Adv进行实验。如图5.16所示，在多目标覆盖任务中，任务目标简单，所有智能体共享奖励，AHAC算法和其基线表现基本一致。多目标收集任务代表了智能体具有共享的奖励反馈的完全合作任务，多指挥者-行动者任务代表了不同的智能体具有不同的奖励反馈的完全合作任务。并且，在多指挥者-行动者任务中，指定智能体的奖励反馈与另一个特定智能体的观察值有关。在这两个任务中，环境中智能体类型增多、数量增加，所以MADDPG算法的训练效果均不理想。在图5.18和图5.19中，AHAC算法比MAAC算法具有更好的学习效果，集中式Critic获得了包含更多隐藏信息的信息向量，说明RNN在处理具有上下文相关的序列信息方面具有优势。由于注意力机制的核心思想为学习从一个查询值到一系列键值-价值对的映射关系。所以，原始数据的嵌入式会影响查询值和每个键值的相似性计算。因此，AHAC算法在多目标收集任务和多指挥者-行动者任务中的训练效果均优于MAAC算法。

值得注意的是，在训练过程中，AHAC算法的Critic网络参数使用正交初始化。经实验验证，正交初始化明显优于随机正态分布初始化。

5.2.4.2 可扩展性

|  |  |
| --- | --- |
| 图5.20 多追捕者-逃逸者（3 vs. 1）任务中，追捕者获取的平均收益 | 图5.21 多追捕者-逃逸者（3 vs. 2）任务中，追捕者获取的平均收益 |
| 图5.22 多追捕者-逃逸者（3 vs. 3）任务中，追捕者获取的平均收益 | 图5.23 多追捕者-逃逸者（6 vs. 2）任务中，追捕者获取的平均收益 |

|  |  |
| --- | --- |
| 图5.24 多追捕者-逃逸者（6 vs. 6）任务中，追捕者获取的平均收益 | 图5.25 多追捕者-逃逸者（9 vs. 3）任务中，追捕者获取的平均收益 |

可扩展性反映了AHAC算法在解决大规模多智能体问题时的性能。通过在多追捕者-逃逸者任务场景中设置不同数量的智能体，测试AHAC算法的可扩展性。在该任务中，逃逸者的策略使用DDPG算法[62]进行训练。如图5.20至图5.25所示，当环境中有更多的追捕者和逃逸者时，与MADDPG算法中非选择性地使用所有信息的Critic网络相比，AHAC算法利用HAM，仍然可以从大规模信息中获取重要信息，有选择地关注特定智能体，将原始信息压缩成一个固定大小的信息向量，具有一定的可扩展性。

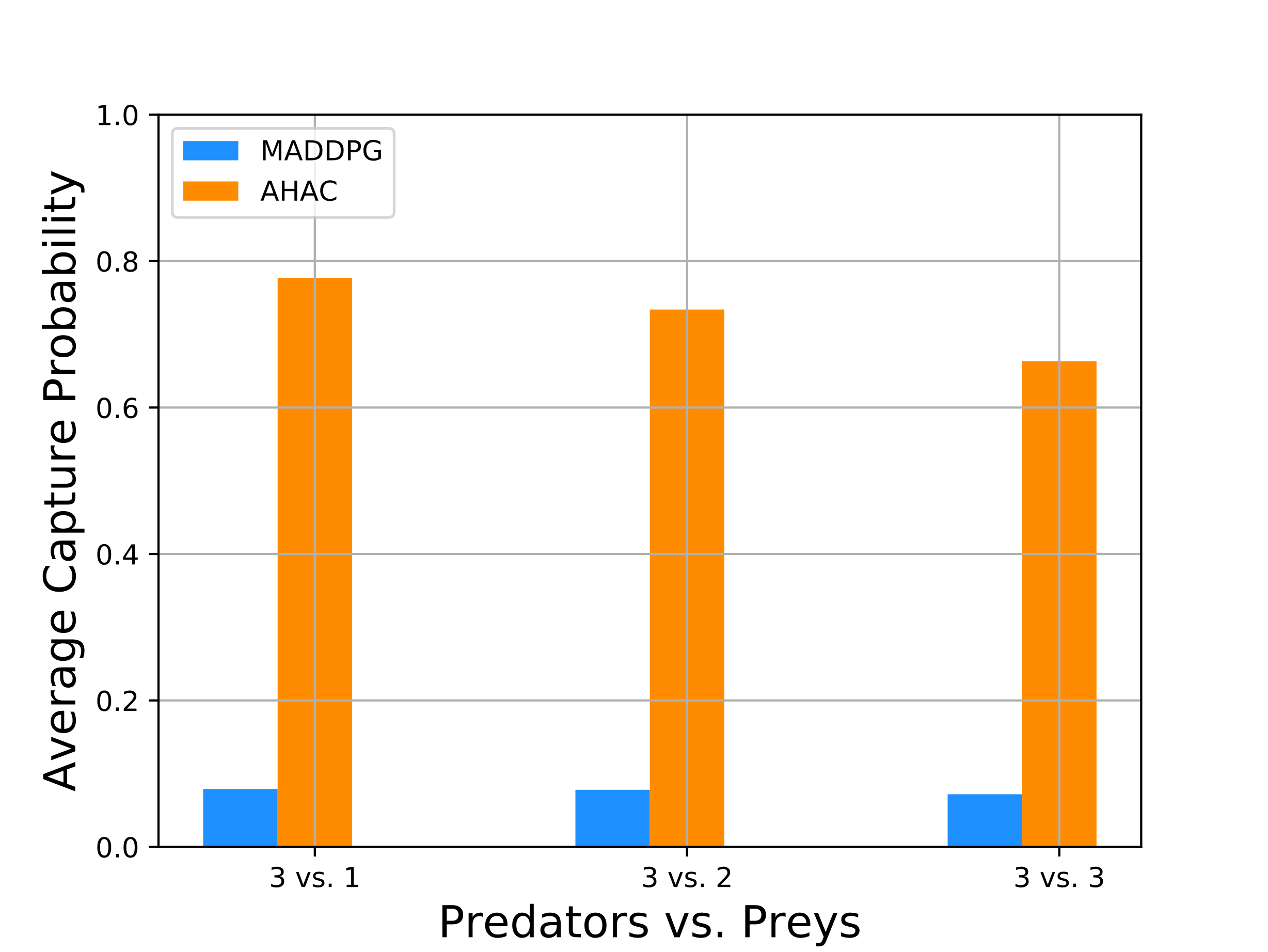


图5.26 在不同数量的逃逸者的情况下，逃逸者被捕捉的概率

在多智能体混合任务中，由于环境中的动态博弈中存在多个纳什均衡，智能体的最优策略往往不是唯一的。在多追捕者-逃逸者任务场景中，我们设定不同数量的逃逸者，对追捕者策略的性能进行测试。由于逃逸者数量的增加，两类智能体之间的互动关系更加复杂多样，追捕者学习到最优策略的难度增加。例如，在多追捕者-逃逸者（3 vs. 1）任务中，环境中只存在一个逃逸者，追捕者的最优策略是唯一的，即所有追捕者追捕唯一的逃逸者；当环境中存在多个逃逸者时，追捕者可以选择与其他追捕者共同行动，也可以选择单独追捕距离较近的逃逸者。如图5.20至图5.22所示，在固定追捕者的数量为3的设定下，增加逃逸者数量，使用AHAC算法训练的追捕者均可以学到较优的策略。

如图5.26所示，每个训练周期内，追捕者平均捕获逃逸者的概率可以更直观地展示追捕者策略的性能。在这种动态任务环境的博弈中，由于使用了HAM，集中式Critic网络可以根据接收到信息更准确地评估Q值，帮助Actor网络学习更优的策略。因此，AHAC算法比MADDPG算法具有更好的训练效果。

5.2.4.3 多头注意力结果的拼接方式

AHAC算法使用了多头注意力，在多个独立的低维特征子空间中并行地计算重要性分布，最后对多个特征子空间的加权结果进行汇总，得到代表其他智能体对智能体共同影响的信息向量，并将其输入到集中式Critic网络中，因此我们期望更好地整合在多个不同的低维特征子空间中的加权信息。与MAAC算法直接对所有注意力头的加权结果进行拼接[84]或Transformer在对所有注意力头的加权结果拼接后执行线性投影不同[78]，AHAC算法在对所有注意力头的加权结果拼接后，对拼接结果使用非线性激活函数再进行一次拟合。AHAC算法对多个注意头的加权结果进行拼接，使用一层带有激活函数的MLP层对拼接结果进行非线性投影，更好地拟合其他智能体对智能体的共同影响。

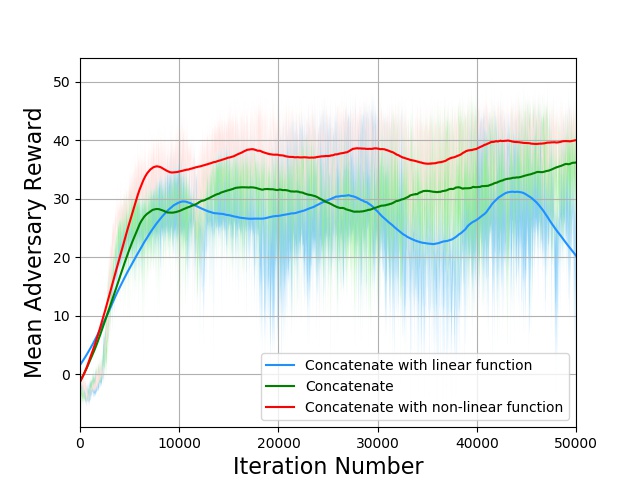


图5.27 在多追捕者-逃逸者（3 vs. 1）任务中，注意力头结果拼接方式对追捕者策略的影响

当注意头数量设定为4时，在多追捕者-逃逸者（3 vs. 1）任务中，AHAC算法分别使用上述所描述的三种方法：直接拼接、拼接后进行线性映射和拼接后进行非线性映射，获取信息向量，实验结果如图5.27所示。显然，拼接后进行非线性映射的方法使得多个注意头的拼接结果具有更强的拟合能力，集中式Critic网络的输入信息更为准确，因此集中式Critic网络可以更好地辅助Actor网络学习策略，使智能体具有更好的性能。

5.2.4.4 注意力头的数量

为了研究注意头数量对HAM计算结果的影响，我们研究了在不同的注意力头的数量的情况下，AHAC算法在多追捕者-逃逸者（3 vs. 1）任务中的表现，追捕者获取的平均收益如图5.28所示。在多头注意力机制中，实际是将数据投影到多个低维特征子空间，即多个注意力头，并在每个注意力头内计算注意力权重。每个注意力头投影后的数据维度通常与注意头的大小（即，特征子空间的维度）有关。一种常见的启发式方法是用计算注意力头的大小。这种启发式最初是在Transformer中提出的，现在已经成为多头注意力的标准启发式。值得注意的是，当注意头的数目设置为1时，AHAC算法实际上采用的是单头注意力机制。



图5.28 在多追捕者-逃逸者（3 vs. 1）任务中，注意力头数量对追捕者策略的影响

通过对比实验结果，当注意头的数量分别设定为1、2和4时，我们发现更多的注意头可以提高训练过程中的稳定性。然而，不同的注意头数对AHAC算法的性能影响不大。但是，根据启发式的计算，增加注意力头的数量，会减少每个注意力头的大小，从而降低单独的注意力头的表达能力。所以，当注意力头数量设置为8时，由于注意力头上的信息损失过多，追捕者的表现显著下降。

5.2.4.5 可视化注意力权重

为了分析基于HAM的集中式Critic中，个体级注意力权重分布和组级注意力权重分布的关系，我们在多追捕者-逃逸者（3 vs.3）任务中，对追捕者的个体级注意权重和组级注意力权重进行可视化，如图5.29所示。

如我们所设想的一样，在个体级注意力计算中获得较高的权重的智能体，其所在的组也会在组级注意力计算中获得更高的权重。在多追捕者-逃逸者任务中，我们设计的奖励反馈是在追捕者之间共享的，因此智能体的权重分布只与智能体之间的相对位置信息有关。当追捕者附近没有待捕获的逃逸者，且其他追捕者距离较近时，追捕者会优先选择避免与其他追捕者发生碰撞，即给予友方智能体更多的关注；当追捕者附近存在逃逸者时，则选择尽可能靠近逃逸者，即给予敌方智能体更多的关注。因此，基于HAM的集中式Critic网络能够有效利用所有智能体中较为重要的信息，计算更为准确的Q值，使Actor网络学习更优的策略，智能体执行的动作可以获得更高的收益。



图5.29 在多追捕者-逃逸者（3 vs. 3）任务中，对个体级和组级的注意力权重可视化

5.3 本章小结

本章主要描述了原型系统架构设计、关键模块的实现以及相关实验。首先，描述了原型系统开发及测试所使用的环境配置。其次，描述了原型系统整体架构设计并对其中的关键模块进行了详细描述，主要包括了算法所需参数和方法以及具体的训练流程。然后，对原型系统的关键模块进行了实验验证和实验分析，验证了MACTP模型和AHAC算法的正确性和有效性，同时对影响性能的因素进行了详细分析。

结 束 语

多目标动态规划任务是智能无人集群的典型任务，可以应用到无人机集群、无人车集群的任务规划中，具有广阔的应用前景。本文以智能无人集群的多目标动态规划任务为背景，设计并实现了一套集任务建模、模型训练为一体的智能无人集群多目标动态规划原型系统。通过该系统，可以有效地解决智能无人集群多目标动态规划问题。首先，该系统对智能无人集群多目标规划问题进行建模，转化为MARL任务，利用人工势场法设计奖励反馈，使用奖励反馈描述任务约束；其次，根据实际需要，针对具有逃逸策略的目标，对MARL算法进行改进，解决当环境中智能体数量较多时，智能体难以学到有效策略的问题，使算法具有更广泛的应用范畴。我们主要对以下三个方面进行了探索。

（1）将智能无人集群的多目标动态规划问题抽象为MAS中的多智能体任务，并使用多智能体Actor-Critic算法框架对智能体进行训练，提高智能体的策略在未知的动态环境中的实时性、鲁棒性和可扩展性。MACTP模型将多目标动态规划问题建模为多智能体任务，利用人工势场法设计奖励反馈并利用奖励反馈描述任务约束，使用MARL算法，在动态环境中对模型进行训练，鼓励智能体充分探索环境，使智能体学习到有效的策略，提升训练后的智能体策略的鲁棒性和可扩展性。

（2）在多目标动态规划任务中，当智能无人集群需要捕获的目标具有逃逸策略时，则该任务则属于混合任务。在混合任务中，无人集群中的智能体在学习策略时，需要同时考虑来自集群中的其他智能体信息和来自待捕获的目标智能体的信息，智能体学习到的策略对接收的信息非常敏感。针对在多智能体Actor-Critic框架中，Critic网络的输入空间将随环境中的智能体数量的增多而快速增长，智能体难以根据Q值反馈学习到有效策略的问题，使用HAM对现有的MARL算法进行改进，对智能体接收的信息进行表示学习和加权计算，压缩为固定长度的信息向量，解决现有的多智能体Actor-Critic框架对智能体数量敏感的问题。AHAC算法不仅可以解决静态目标的多目标规划问题，而且可以解决目标是具有逃逸策略的智能体的多目标规划问题，提高了算法的可扩展性。

（3）针对传统智能无人集群多目标动态规划算法在动态环境中计算难度大，智能体的策略难以实时应对未知环境的问题，设计并实现基于MARL的多目标动态规划原型系统。该原型系统通过构建MAS，设计奖励反馈描述多目标动态规划任务的约束条件，利用多智能体Actor-Critic框架对模型进行训练，保存后的模型即为智能体的策略。该原型系统充分发挥了MARL的优点，针对多目标动态规划任务，训练后的智能体可以适应未知的动态环境，根据自身的策略做出决策，提高了智能体策略的实时性和鲁棒性，并且具有一定的可扩展性。同时，针对目标是具有逃逸策略的智能体，环境中智能体数量增多，Critic网络的输入空间维度快速增长的问题，该原型系统采用基于HAM的MARL算法，利用HAM对所有智能体的信息进行加权压缩，提高了系统的可扩展性，提升了智能体的性能，不仅可以解决静态目标的多目标规划问题，并且可以适用于动态目标的复杂情况。

我们使用MARL算法解决智能无人集群的多目标动态规划任务，为该任务提供了新的研究思路。但是，本文的研究工作仍然存在可以进一步地深入研究和改进的地方。基于本文上述的研究内容基础，未来的研究内容主要包括以下三个方面。

（1）我们提出的基于HAM的MARL算法对所有智能体的信息计算了两次重要性分布，当智能体数量增多时，训练时间会显著增加，如何优化网络结构，降低结构冗余性，减少不必要的重复性计算，是我们未来研究内容之一。

（2）我们设计的基于MARL的多目标动态规划原型系统缺乏三维仿真实验。该原型系统虽然通过对比实验，验证了该系统的正确性与有效性，但是考虑到直接将模型部署到实物进行实验的成本及安全性，需要首先对原型系统进行三维仿真实验，模拟该原型系统在无人集群中的应用。如何将该原型系统迁移到Gazebo等三维物理仿真实验平台是本文的另外一个重要研究内容。

（3）本文采用的MARL算法均使用随机采样的方法，从经验回放池中批次抽取经验对智能体进行训练，随机采样使训练所用的经验样本满足独立同分布条件，但是缺乏对关键状态的关注。设计重要性采样的规则，提高训练效率，增快智能体的学习速度也是未来的研究内容之一。

致 谢

时光转瞬即逝，心中百感交集。经过两年半的学习，自己变得更加成熟，并且找到了自己的研究方向。感谢所有伴我成长的老师、家人以及朋友，让我的研究生生活留下温暖的回忆。

由衷感谢我的指导老师史殿习研究员在我读研期间对我的指导。感谢史老师每周仔细审阅我们的工作周报，并且定期开展组会，听取近期科研进展汇报，针对研究课题提出宝贵意见，确定研究方向、确立研究思路。同时，史老师也非常关心我们的日常生活，尽心尽力的为我们解决困难。

衷心感谢军事科学院和天津人工智能创新中心的薛超老师、金松昌老师、康颖老师，帮助我加深了对课题的理解，及时确立了研究目标，顺利克服了研究难点，为顺利完成毕业论文奠定了坚实的基础。

衷心感谢蔡巍大队长、姚佳政委、王维教导员、周兵队长等领导，谢谢你们在大队日常管理上的付出，为日常生活提供了保障。。

感谢王柳静、潘晨、伽晗、李若祥、王明坤等同门师姐、师兄，给予我学术和人生方向的建议，帮助我顺利开始我的研究生生涯。感谢姜浩、王功举、宫鹏、于晓晓、孙若琳、张耀文、佘馥江、徐天齐、王宁、王弢、邹钧博同学和赵琛然等同门师妹、师弟，很幸运遇到你们，在科研学习以及日常生活中，相互帮助、相互督促、共同进步，为我提供了诸多帮助和支持。

感谢张昆、霍曼妍、王云龙等在科大相识的同学，丰富了我的日常生活，带我感受了长沙这座城市的魅力。

感谢我的父母，谢谢父母对我无私地奉献与付出，让我可以勇敢去追寻自己的人生。

最后，特别感谢百忙中负责评阅我论文的老师们，老师们辛苦了！

参考文献

1. Simon H A. The shape of automation for men and management[M]. New York: Harper & Row, 1965.
2. Rumelhart D E, Hinton G E, Williams R J. Learning representations by back-propagating errors[J]. nature, 1986, 323(6088): 533-536.
3. Bush R R, Mosteller F. Stochastic models for learning[J]. 1955.
4. Mnih V, Kavukcuoglu K, Silver D, et al. Playing atari with deep reinforcement learning[J]. arXiv preprint arXiv:1312.5602, 2013.
5. Mnih V, Kavukcuoglu K, Silver D, et al. Human-level control through deep reinforcement learning[J]. nature, 2015, 518(7540): 529-533.
6. Mordatch I, Abbeel P. Emergence of Grounded Compositional Language in Multi-Agent Populations.[J]. arXiv: Artificial Intelligence, 2017.
7. Russell S J, Norvig P. Artificial intelligence: a modern approach[M]. Malaysia; Pearson Education Limited,, 2016.
8. 李杨, 徐峰, 谢光强, 等. 多智能体技术发展及其应用综述[J]. 计算机工程与应用, 2018, 54(9): 13-21.
9. 张悦. 多智能体深度强化学习方法及应用研究[D].
10. 吴锋. 基于决策理论的多智能体系统规划问题研究[D]. 中国科学技术大学, 2011.
11. Sutton R S, Barto A G. Reinforcement learning: An introduction[M]. MIT press, 2018.
12. Samuel A L. Some studies in machine learning using the game of checkers[J]. IBM Journal of research and development, 1959, 3(3): 210-229.
13. Minsky M L. Theory of neural-analog reinforcement systems and its application to the brain model problem[M]. Princeton University., 1954.
14. Waltz M, Fu K. A heuristic approach to reinforcement learning control systems[J]. IEEE Transactions on Automatic Control, 1965, 10(4): 390-398.
15. Bellman R. A Markovian decision process[J]. Journal of mathematics and mechanics, 1957: 679-684.
16. Hernandez-Leal P, Kartal B, Taylor M E. Is multiagent deep reinforcement learning the answer or the question? A brief survey[J]. learning, 2018, 21: 22.
17. 杜威, 丁世飞. 多智能体强化学习综述[J]. 计算机科学, 2019(8).
18. Buşoniu L, Babuška R, De Schutter B. Multi-agent reinforcement learning: An overview[M]//Innovations in multi-agent systems and applications-1. Springer, Berlin, Heidelberg, 2010: 183-221.
19. LeCun Y, Bottou L, Bengio Y, et al. Gradient-based learning applied to document recognition[J]. Proceedings of the IEEE, 1998, 86(11): 2278-2324.
20. Collobert R, Weston J, Bottou L, et al. Natural language processing (almost) from scratch[J]. Journal of machine learning research, 2011, 12(ARTICLE): 2493− 2537.
21. Krizhevsky A, Sutskever I, Hinton G E. Imagenet classification with deep convolutional neural networks[C]//Advances in neural information processing systems. 2012: 1097-1105.
22. LeCun Y, Bengio Y, Hinton G. Deep learning[J]. nature, 2015, 521(7553): 436-444.
23. 段海滨, 邱华鑫, 陈琳, 等. 无人机自主集群技术研究展望[J]. 科技导报, 2018, 36(21): 90-98.
24. Lawler E L. The quadratic assignment problem[J]. Management science, 1963, 9(4): 586-599.
25. Lloyd S P, Witsenhausen H S. Weapons allocation is NP-complete[C]//1986 Summer Computer Simulation Conference. 1986: 1054-1058.
26. Sayin A, Cherniakov M, Antoniou M. Passive radar using Starlink transmissions: A theoretical study[C]//2019 20th International Radar Symposium (IRS). IEEE, 2019: 1-7.
27. 王祥科, 刘志宏, 丛一睿,等. 小型固定翼无人机集群综述和未来发展[J]. 航空学报, 2020, 041(004):15-40.
28. 唐毅, 司维钊, 王怀超. 无人机集群技术的民用发展[J]. 科教导刊-电子版（上旬）, 2018, 000(008):251-252.
29. 罗德林, 徐扬, 张金鹏. 无人机集群对抗技术新进展[J]. 科技导报, 2017(07):26-31.
30. 袁成. 美国国防高级研究计划局"小精灵"项目[J]. 兵器知识, 2016(9):37-39.
31. 杨王诗剑. 引领海战革命——浅析无人机"蜂群战术"[J]. 兵器知识, 2016(3期):60-63.
32. 马巧云. 基于多Agent系统的动态任务分配研究[D]. 华中科技大学, 2006.
33. 蔡怀平, 陈英武. 武器-目标分配(WTA)问题研究进展[J]. 火力与指挥控制, 2006, 31(012):11-15.
34. Fujita, Satoshi, Masukawa, Masayuki, Tagashira, Shigeaki. A fast branch-and-bound algorithm with an improved lower bound for solving the multiprocessor scheduling problem[J]. 2002.
35. 苏菲, 陈岩, 沈林成. 基于蚁群算法的无人机协同多任务分配[J]. 航空学报, 2008, 029(B05):184-191.
36. Wu F , Zilberstein S , Chen X . Point-based policy generation for decentralized POMDPs[C]// AAMAS. 2010.
37. O'Rourke K P, Carlton W B, Bailey T G, et al. Dynamic routing of unmanned aerial vehicles using reactive tabu search[J]. Military Operations Research, 2001: 5-30.
38. Secrest B R. Traveling salesman problem for surveillance mission using particle swarm optimization[R]. AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOOL OF ENGINEERING AND MANAGEMENT, 2001.
39. 潘斌斌. 多目标路径规划问题的算法综述[J]. 重庆工商大学学报:自然科学版, 2012, 29(5).
40. Hu Z. A container multimodal transportation scheduling approach based on immune affinity model for emergency relief[J]. Expert Systems With Applications, 2011, 38(3): 2632-2639.
41. Androutsopoulos K N , Zografos K G . Solving the multi-criteria time-dependent routing and scheduling problem in a multimodal fixed scheduled network[J]. European Journal of Operational Research, 2009, 192(1):18-28.
42. Tan K C , Chew Y H , Lee L H . A hybrid multi-objective evolutionary algorithm for solving truck and trailer vehicle routing problems[J]. European Journal of Operational Research, 2006, 172(3):855-885.
43. Samanlioglu F , Jr W G F , Kurz M E . A memetic random-key genetic algorithm for a symmetric multi-objective traveling salesman problem[J]. Computers & Industrial Engineering, 2008, 55(2):439-449.
44. Turpin M , Mohta K , Michael N , et al. Goal assignment and trajectory planning for large teams of interchangeable robots[J]. Autonomous Robots, 2014, 37(4):401-415.
45. Qie H , Shi D , Shen T , et al. Joint Optimization of Multi-UAV Target Assignment and Path Planning Based on Multi-Agent Reinforcement Learning[J]. IEEE Access, 2019, 7:146264-146272.
46. Watkins C J C H, Dayan P. Q-learning[J]. Machine learning, 1992, 8(3-4): 279-292.
47. 李江红. 不确定MARKOV决策过程的自适应决策[D]. 上海交通大学, 2000.
48. Hausknecht M, Stone P. Deep recurrent q-learning for partially observable mdps[J]. arXiv preprint arXiv:1507.06527, 2015.
49. Williams R J. Simple statistical gradient-following algorithms for connectionist reinforcement learning[J]. Machine learning, 1992, 8(3-4): 229-256.
50. Sutton R S, McAllester D A, Singh S P, et al. Policy gradient methods for reinforcement learning with function approximation[C]//Advances in neural information processing systems. 2000: 1057-1063.
51. Konda V R, Tsitsiklis J N. Actor-critic algorithms[C]//Advances in neural information processing systems. 2000: 1008-1014.
52. Mnih V, Badia A P, Mirza M, et al. Asynchronous methods for deep reinforcement learning[C]//International conference on machine learning. 2016: 1928-1937.
53. 仲宇, 顾国昌, 张汝波. 多智能体系统中的分布式强化学习研究现状[D]. 2003.
54. Foerster J, Farquhar G, Afouras T, et al. Counterfactual multi-agent policy gradients[J]. arXiv preprint arXiv:1705.08926, 2017.
55. Schulman J, Wolski F, Dhariwal P, et al. Proximal policy optimization algorithms[J]. arXiv preprint arXiv:1707.06347, 2017.
56. Ardi T , Tambet M , Dorian K , et al. Multiagent Cooperation and Competition with Deep Reinforcement Learning[J]. Plos One, 2017, 12(4):e0172395.
57. Raghu M, Irpan A, Andreas J, et al. Can Deep Reinforcement Learning Solve Erdos-Selfridge-Spencer Games?[J]. arXiv: Artificial Intelligence, 2017.
58. Sukhbaatar S, Szlam A, Fergus R, et al. Learning multiagent communication with backpropagation[C]. neural information processing systems, 2016: 2252-2260.
59. Peng P , Wen Y , Yang Y , et al. Multiagent Bidirectionally-Coordinated Nets: Emergence of Human-level Coordination in Learning to Play StarCraft Combat Games[J]. 2017.
60. Schuster M , Paliwal K K . Bidirectional recurrent neural networks[J]. IEEE Transactions on Signal Processing, 1997, 45(11):2673-2681.
61. Lowe R, Wu Y I, Tamar A, et al. Multi-agent actor-critic for mixed cooperative-competitive environments[C]//Advances in neural information processing systems. 2017: 6379-6390.
62. Lillicrap T P, Hunt J J, Pritzel A, et al. Continuous control with deep reinforcement learning[J]. arXiv preprint arXiv:1509.02971, 2015.
63. Albrecht S V , Stone P . Autonomous Agents Modelling Other Agents: A Comprehensive Survey and Open Problems[J]. Artificial Intelligence, 2017, 258(MAY):66-95.
64. He H , Boyd-Graber J , Kwok K , et al. Opponent Modeling in Deep Reinforcement Learning[J]. 2016.
65. Li S, Wu Y, Cui X, et al. Robust Multi-Agent Reinforcement Learning via Minimax Deep Deterministic Policy Gradient[C]. national conference on artificial intelligence, 2019, 33(01): 4213-4220.
66. Morimoto J , Doya K . Robust reinforcement learning.[J]. Neural Computation, 2005.
67. Pinto L, Davidson J, Sukthankar R, et al. Robust adversarial reinforcement learning[C]. international conference on machine learning, 2017: 2817-2826.
68. Silver D, Huang A, Maddison C J, et al. Mastering the game of Go with deep neural networks and tree search[J]. nature, 2016, 529(7587): 484.
69. 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.
70. Gu S, Holly E, Lillicrap T, et al. Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates[C]//2017 IEEE international conference on robotics and automation (ICRA). IEEE, 2017: 3389-3396.
71. Shalev-Shwartz S, Shammah S, Shashua A. Safe, multi-agent, reinforcement learning for autonomous driving[J]. arXiv preprint arXiv:1610.03295, 2016.
72. Ng A Y, Harada D, Russell S, et al. Policy Invariance Under Reward Transformations: Theory and Application to Reward Shaping[C]. international conference on machine learning, 1999: 278-287.
73. Wang C, Wang J, Shen Y, et al. Autonomous Navigation of UAVs in Large-Scale Complex Environments: A Deep Reinforcement Learning Approach[J]. IEEE Transactions on Vehicular Technology, 2019, 68(3): 2124-2136.
74. Mnih V, Heess N, Graves A. Recurrent models of visual attention[C]//Advances in neural information processing systems. 2014: 2204-2212.
75. Chaudhari S, Polatkan G, Ramanath R, et al. An attentive survey of attention models[J]. arXiv preprint arXiv:1904.02874, 2019.
76. Cho K, Van Merriënboer B, Bahdanau D, et al. On the properties of neural machine translation: Encoder-decoder approaches[J]. arXiv preprint arXiv:1409.1259, 2014.
77. Xu K, Ba J, Kiros R, et al. Show, attend and tell: Neural image caption generation with visual attention[C]//International conference on machine learning. 2015: 2048-2057.
78. Vaswani A, Shazeer N, Parmar N, et al. Attention is all you need[C]//Advances in neural information processing systems. 2017: 5998-6008.
79. Yang Z, Yang D, Dyer C, et al. Hierarchical attention networks for document classification[C]//Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: human language technologies. 2016: 1480-1489.
80. Littman M L. Markov games as a framework for multi-agent reinforcement learning[M]//Machine learning proceedings 1994. Morgan Kaufmann, 1994: 157-163.
81. Silver D, Lever G, Heess N, et al. Deterministic policy gradient algorithms[C]. 2014.
82. 程玉虎. 连续状态-动作空间下强化学习方法的研究[J]. 2005.
83. Haarnoja T, Zhou A, Abbeel P, et al. Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement[J].
84. Iqbal S, Sha F. Actor-attention-critic for multi-agent reinforcement learning[C]//International Conference on Machine Learning. PMLR, 2019: 2961-2970.
85. Brockman G , Cheung V , Pettersson L , et al. OpenAI Gym[J]. 2016.
86. Jang E, Gu S, Poole B. Categorical reparameterization with gumbel-softmax[J]. arXiv preprint arXiv:1611.01144, 2016.
87. Zaremba W, Sutskever I, Vinyals O. Recurrent neural network regularization[J]. arXiv preprint arXiv:1409.2329, 2014.

作者在学期间取得的学术成果

1. Yajie Wang, Dianxi Shi, Chao Xue, Hao Jiang, Gongju Wang, Peng Gong. “AHAC : Actor Hierarchical Attention Critic for Multi-Agent Reinforcement Learning.” 2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC).