

# 强化学习驱动的海战场多智能体协同作战仿真算法

石鼎, 燕雪峰, 宫丽娜, 张静宣, 关东海, 魏明强\*

(南京航空航天大学 计算机科学与技术学院, 江苏 南京 211100)

**摘要:** 未来海战场形势瞬息万变, 亟需依托人工智能技术实现对海战场环境的高质量作战仿真, 以全面优化和提升我军战斗力, 达成克敌制胜的目的。作战单元的协同合作是实现海战场作战仿真的关键环节, 如何实现多智能体之间的均衡决策是作战仿真首要解决的问题。基于解耦的优先经验回放机制和注意力机制, 提出强化学习驱动的多智能体协同作战仿真算法(*multi-agent reinforcement learning-based cooperative combat simulation, MARL-CCSA*)。在 *MARL-CCSA* 基础上, 利用专家经验, 设计一种多尺度奖励函数, 并基于此函数构建一个海战场作战仿真环境, 使 *MARL-CCSA* 在此环境中训练易于收敛。设计想定进行仿真实验, 并与其他算法的效果进行对比, 验证 *MARL-CCSA* 的可行性与实用性。

**关键词:** 作战仿真; 协同工作; 强化学习; 优先经验回放; 注意力机制; 多尺度奖励函数

中图分类号: TP391.9

文献标志码: A

文章编号: 1004-731X(2023)04-0786-11

DOI: 10.16182/j.issn1004731x.joss.21-1321

**引用格式:** 石鼎, 燕雪峰, 宫丽娜, 等. 强化学习驱动的海战场多智能体协同作战仿真算法[J]. 系统仿真学报, 2023, 35(4): 786-796.

**Reference format:** Shi Ding, Yan Xuefeng, Gong Lina, et al. Multi-agent Cooperative Combat Simulation in Naval Battlefield with Reinforcement Learning[J]. Journal of System Simulation, 2023, 35(4): 786-796.

## Multi-agent Cooperative Combat Simulation in Naval Battlefield with Reinforcement Learning

Shi Ding, Yan Xuefeng, Gong Lina, Zhang Jingxuan, Guan Donghai, Wei Mingqiang\*

(School of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing 211100, China)

**Abstract:** Due to the rapidly-changed situations of future naval battlefields, it is urgent to realize the high-quality combat simulation in naval battlefields based on artificial intelligence to comprehensively optimize and improve the combat effectiveness of our army and defeat the enemy. The collaboration of combat units is the key point and how to realize the balanced decision-making among multiple agents is the first task. Based on decoupling priority experience replay mechanism and attention mechanism, a multi-agent reinforcement learning-based cooperative combat simulation (*MARL-CCSA*) network is proposed. Based on the expert experience, a multi-scale reward function is designed, on which a naval battlefield combat simulation environment is constructed. The proposed multi-scale reward function could speed the convergence of multiple agents. The feasibility and practicability of *MARL-CCSA* is verified by the simulation experiment and the comparison with the other methods.

**Keywords:** combat simulation; collaboration; reinforcement learning; prioritized experience replay; attention mechanism; multi-scale reward function

收稿日期: 2021-12-20 修回日期: 2022-03-01

基金项目: 国家自然科学基金面上项目(62172218)

第一作者: 石鼎(1996-), 男, 硕士生, 研究方向为多智能体强化学习。E-mail: shiding0614@163.com

通讯作者: 魏明强(1985-), 男, 教授, 博士, 研究方向为军事智能。E-mail: mqwei@nuaa.edu.cn

## 0 引言

党的十九大报告强调“加快军事智能化发展, 提高基于网络信息体系的联合作战能力、全域作战能力”。未来的战争将会是智能信息化大规模联合作战, 军事智能化已经成为新一轮军事变革的核心驱动力, 从根本上改变了未来战争取得胜利的方式、军力构成和战争形态。

机器学习等高新智能技术已广泛应用于海战“作战样式、特点和制胜机理”分析之中, 并催生出“跨域协同、敏捷指挥和精准控制”的海战新形态, 即智能化海战。通过将人工智能技术广泛运用于海上军事活动中, 实现“以智赋能、以智释能”, 从而全面优化我军作战方案, 提升我军战斗力, 最终达到克敌制胜的目的。

随着现代战争的规模和复杂程度日益加深, 传统的人工决策方式已经不能适用于瞬息万变的海战场环境。未来的战争模式越来越趋向于智能化和无人化, 在瞬息万变的战场环境中, 需要根据战场态势自主进行战场决策, 采取作战行动, 完成作战任务, 开拓军队在各种极限环境下的作战能力。

战场推演是军事决策的重要手段, 如何将人工智能技术运用到战场推演中, 是未来军事智能化的发展趋势。目前, 以机器学习为代表的人工智能技术已经取得突破性进展, 在许多领域的表现已经远远超过了人类。但是由于军事活动存在特殊性, 人工智能技术在军事仿真领域还没有取得显著成果, 一方面是由于军事场景的高复杂度, 使得难以通过军事仿真环境还原战场环境; 另一方面, 军事数据不仅具有数据规模庞大、内容种类繁多、价值密度较低等典型的非军事数据特征, 还具有“一超一高一强”的特性, 即超复杂性(数据维度庞大, 数据关系复杂, 非结构化数据增多)、高安全性(更加严重的安全威胁, 例如遭到侦察窃取、产生系统漏洞或其他的能够造成泄密失密的打击手段)、强对抗性(数据获取与反爬措施相

互对抗, 真假数据错综复杂, 要求对数据真伪有极强的辨别能力), 限制了监督和半监督机器学习在军事场景中的应用。

随着深度学习和强化学习的发展, 深度强化学习的出现让人工智能技术在军事仿真方面的应用有了新的发展方向。目前的军事仿真技术大多使用决策树等传统逻辑结构, 通过仿真软件内部的逻辑进行军事仿真。然而这种仿真并不能真正地还原战场环境, 会受到人工逻辑设计的影响, 不能算是真正意义上的军事仿真。所以亟需将人工智能技术尤其是深度强化学习技术应用到军事仿真领域。

多智能体强化学习算法非常适合应用于小规模场景的战场推演, 以 MADDPG(multi-agent deep deterministic policy gradient)<sup>[1]</sup>为代表的多智能体强化学习算法在一些仿真实验中已经取得了一定的效果。MADDPG 是对 DDPG(deep deterministic policy gradient)算法<sup>[2]</sup>进行的改进和扩展, 以应用于多智能体环境中。MADDPG 算法的核心思想是每个智能体的 Critic 部分能够获取其余智能体的动作、奖励、观测等信息。算法的结构是中心化训练去中心化执行, 在训练阶段, 使用一个全局的 Critic 来获取所有智能体的信息, 从而获取了全局信息, 并以此来引导每个独立的智能体进行训练; 在测试阶段, 丢弃全局 Critic, 每个智能体只利用自己观测到的局部信息来采取行动。但是军事环境往往比较复杂, 智能体个数比较多, 使用常规的 MADDPG 算法训练模型, 收敛速度通常比较缓慢, 收敛不够稳定, 而且如果环境的奖励函数设置不好, 容易出现模型不收敛的情况。

本文首先基于 MADDPG 算法提出 2 种改进方案: ①在 MADDPG 模型原有的全局 Critic 网络基础上, 为每个智能体添加单独的局部 Critic 网络, 有效应对多智能体间的数据耦合问题, 增强模型的稳定性, 同时将 MADDPG 算法的经验回放机制改进为优先经验回放机制, 有效增强模型的收敛速度; ②在 Actor-Critic 网络中添加注意力模块,

让 Actor-Critic 网络更加关注与该智能体相关的态势特征,改善智能体的行为,增强模型的性能。其次,本文针对军事仿真场景设计一种多尺度奖励函数,将专家经验嵌入到奖励函数中,并根据该奖励函数构建一个军事仿真环境进行模型训练,验证本文算法模型的收敛性。然后,将本文的效果与 MADDPG 算法和 QMIX 算法<sup>[3]</sup>的效果进行对比,验证本文算法的实用性。最后,进行消融实验,验证本文提出改进方案的有效性。

## 1 多智能体强化学习简述

### 1.1 强化学习

强化学习讨论的问题是智能体怎么样在一个复杂的环境中去极大化它能获得的奖励。如图1所示,在强化学习过程中,智能体和环境一直在交互,智能体获取到环境的状态,然后智能体会利用获取的状态进行决策,即输出一个动作。动作会对环境产生影响,环境会返回给智能体当前动作的奖励,同时会更新环境的状态。智能体再与环境交互获取环境的下一个状态。循环往复,直到环境判断这次交互结束。智能体的目标就是改善决策方式,从而尽可能多地从环境中获取奖励。

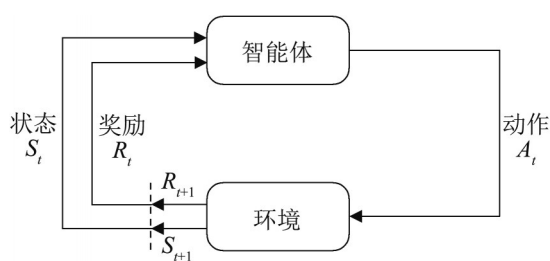


图1 强化学习过程  
Fig. 1 Procedure of reinforcement learning

对于一个强化学习的智能体,它的决策方式可能是基于价值的,可能是基于策略的,也可能是二者相结合。基于价值的决策方式常用于解决离散的状态空间和离散的动作空间的强化学习问题。由于状态空间和动作空间都是离散的,所以

智能体通过价值函数来评判每一个状态的价值,同时也评判每一个状态下选取某个动作的价值,这个价值是未来收益的总和。基于价值的决策方式的代表性算法有 Q-learning<sup>[4]</sup>、Sarsa<sup>[5]</sup>等。基于策略的决策方式通过输入某一个状态,直接输出这个状态应该做出的动作,这种决策方式有2类:①随机性策略,它的输出是所有动作的一个概率,然后根据这个概率进行采样;②确定性策略,只输出最有可能的动作。基于策略决策方式的代表性算法有 Policy Gradient<sup>[6]</sup>。

神经网络出现后,为了解决连续的状态空间的强化学习问题,DeepMind 将神经网络与 Q-learning 算法相结合,提出了 DQN(deep Q network)<sup>[7]</sup>算法;为了解决连续的动作空间的强化学习问题,Lillicrap 等提出了深度确定性策略梯度(deep deterministic policy gradient, DDPG)算法,该算法结合了 DQN 算法和 Actor-Critic<sup>[8]</sup>结构,解决了 DQN 无法应对连续动作空间的难题,是深度学习在连续动作空间问题上的重大发展。

### 1.2 多智能体强化学习

随着强化学习的应用场景越来越复杂,出现了越来越多的多智能体应用场景,多智能体强化学习也得到了快速发展。多智能体强化学习分为完全去中心化、完全中心化、中心化训练去中心化执行3种架构,Hernandez-Leal 等<sup>[9]</sup>将多智能体强化学习分为行为分析、通信学习、协作学习和智能体建模四类<sup>[10-15]</sup>。

在完全去中心化的架构中,每个智能体独立和环境进行交互,仅仅使用自己的观测和奖励值来更新策略,智能体之间不进行交流。这种架构本质上属于将单智能体的方法应用到多智能体的场景中,效果往往较差。

在完全中心化的架构中,所有智能体将自己观测到的信息传送给中央控制器,以便其掌握所有智能体的观测、动作和奖励。智能体上没有决策网络,其不负责决策,决策均由中央控制器控



制, 智能体只负责执行指令。这种架构往往效率不高, 中央控制器要收集到所有智能体的信息之后才能进行决策, 所有智能体都要等待最慢的那一个传送信息后才能获取中央控制器的决策。

在中心化训练去中心化执行的架构中, 每个智能体都有自己的策略网络用于决策, 而中央控制器中存在一个全局 Critic 网络, 用于对每个智能体的决策进行打分。在训练时, 使用中央控制器来训练每个智能体, 训练结束后便不再需要中央控制器, 每个智能体能够根据自己的观测进行决策。这种架构既能够保证每个智能体之间的交流信息, 又能够保证执行的效率。目前较为常用的多智能体算法几乎都属于这种架构。

## 2 MARL-CCSA 算法

### 2.1 MADDPG 算法介绍

多智能体强化学习最大的问题在于, 所有智能体都在更新策略, 因此, 对于某一个独立的智能体来说, 环境并不是稳定的。即使是已经更新到能够获取最大奖励值的策略, 只要其他智能体的策略发生改变, 那么获取最大奖励值的策略也会随之改变。针对上述问题, MADDPG 采取中心化训练去中心化执行的结构, 对传统的 Actor-Critic 算法进行了一个改进。在 MADDPG 算法中, Critic 会获取所有智能体的信息, 对所有智能体的策略进行融合, 进而引导每个独立智能体进行学习, 让每个智能体都能考虑到其他智能体的策略带来的影响。因此, 需要一些额外的全局信息进行学习, 需要在执行时使用局部信息进行决策。

在 MADDPG 算法中, 若用  $\theta=(\theta_1, \theta_2, \dots, \theta_n)$  表示  $n$  个智能体策略的参数,  $\pi=(\pi_1, \pi_2, \dots, \pi_n)$  表示  $n$  个智能体的策略。针对第  $i$  个智能体的累积期望奖励可以表示为

$$J(\theta_i)=E_{s \sim p^\pi, a_i \sim \pi_{\theta_i}} \left( \sum_{t=0}^{\infty} \gamma^t r_{i,t} \right) \quad (1)$$

式中:  $p^\pi$  为状态分布;  $\pi_{\theta_i}$  为动作分布;  $\gamma$  为折扣因

子;  $r_{i,t}$  为  $t$  时刻智能体的奖励。对于随机策略, 求策略梯度为

$$\nabla_{\theta_i} J(\theta_i)=E_{s \sim p^\pi, a_i \sim \pi_{\theta_i}} [\nabla_{\theta_i} \log \pi_{\theta_i}(a_i|o_i) Q_i^\pi(\mathbf{x}, a_1, a_2, \dots, a_n)] \quad (2)$$

式中:  $o_i$  为第  $i$  个智能体观测;  $a_i$  为对应的动作;  $\mathbf{x}=[o_1, \dots, o_n]$  为观测向量;  $Q_i^\pi(\mathbf{x}, a_1, a_2, \dots, a_n)$  为第  $i$  个智能体综合状态-动作价值函数。由于每个智能体的目的不同, 每个智能体独立学习自己的  $Q_i^\pi$  函数, 所以每个智能体的奖励函数也各不相同。智能体之间通过全局 Critic 的调节, 从而能够完成合作或竞争任务。由于脱胎于 DDPG 算法, 因此动作空间可以是连续的。

式(2)为随机策略梯度算法, 将其拓展到确定性策略, 梯度公式为

$$\nabla_{\theta_i} J(\mu_i)=E_{\mathbf{x}, a \sim D} [\nabla_{\theta_i} \mu_i(a_i|o_i) \nabla_{a_i} Q_i^\mu(\mathbf{x}, a_1, a_2, \dots, a_n)]_{a_i=\mu_i(o_i)} \quad (3)$$

式中:  $\mu_i$  为  $N$  个连续的策略;  $D$  为经验回放缓冲区。集中式的 Critic 的更新方法借鉴了 DQN 中 TD (temporal difference) 与目标网络思想。

式(4)和(5)表示 MADDPG 算法的预测值和真实值, 利用预测值和真实值之间的差异来更新算法:

$$L(\theta_i)=E_{\mathbf{x}, a, r, \mathbf{x}'} [(Q_i^\mu(\mathbf{x}, a_1, a_2, \dots, a_n) - y)^2] \quad (4)$$

$$y=r_i+\gamma \bar{Q}_j^{\mu'}(\mathbf{x}', a'_1, a'_2, \dots, a'_n) | a'_j=\mu'_j(o_j) \quad (5)$$

式中:  $\bar{Q}_j^{\mu'}$  为目标网络,  $\mu'=(\mu'_1, \mu'_2, \dots, \mu'_n)$  为目标策略具有滞后更新的参数  $\theta'_j$ 。可以通过拟合函数逼近的方式得到其他智能体的策略, 而不需要通信交互。

### 2.2 MARL-CCSA 改进策略

#### 2.2.1 解耦的优先经验回放机制

在 MADDPG 算法中, 智能体会将动作传输到中央控制器。由于中央控制器中只有一个 Critic, 要对每个智能体的行为打分, 在更新这个 Critic 时, 需要使用所有智能体的奖励来进行更新。这样的模型虽然可以让多智能体之间通信, 但同时也加强了多智能体间的数据耦合。

为了解决多智能体间的数据耦合问题,本文提出 MADDPG 算法解耦机制 (decoupling mechanism)。对于每个智能体,都让一个额外的 Critic 网络给输出的动作打分,即对于每一个智能体,输出的动作不仅仅传输给中央控制器,同时传输到和该智能体匹配的 Critic 网络。所有智能体匹配的 Critic 网络采用相同的网络结构,但是彼此不通信,只用于更新每个单独的智能体<sup>[16]</sup>。

MADDPG 同样使用了经验回放机制,经验回放机制通常从经验缓冲池中均匀采样经验。虽然这种采样机制在一些算法中取得了不错的效果,但是这种采样机制忽略了数据样本之间的重要性差异,导致采样的效率不高,模型收敛速度变慢。Schaul 等<sup>[17]</sup>提出了单智能体的优先经验回放技术,解决均匀采样的问题。该技术引入了一个 TD-error 给每个数据样本进行重要性标记,在每次采样更新策略网络时,根据此次采样获取梯度的大小给该样本标记。使用随机优先采样法,根据 TD-error 的大小指定采样策略,设置重要的样本采样的几率更大,不重要的样本采样的几率更小。本文将单智能体的优先采样机制引入到多智能体领域,对于多智能体系统中的每一个智能体,在采样的时候都进行重要性标记,实时更新样本的重要性权重。

本文的 Critic 网络沿用 MADDPG 算法中的网络结构,为输入态势特征数 $\times 64 \times 64 \times 1$ 的全连接神经网络。本文将 MADDPG 算法解耦机制和优先经验回放机制相结合,其算法流程如图2所示。

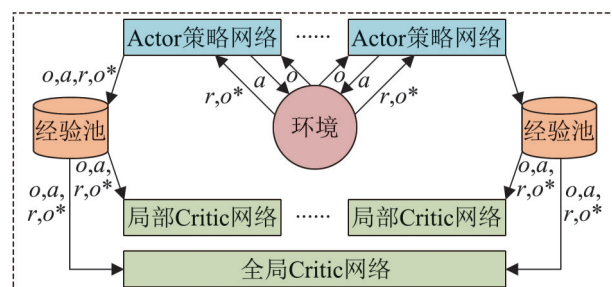


图2 解耦与经验回放机制

Fig. 2 Decoupling and empirical playback mechanism

## 2.2.2 注意力 Actor-Critic 网络

Attention 机制<sup>[18]</sup>广泛应用于自然语言处理和计算机视觉领域,常用的结构是通道注意力 SE (Squeeze-and-Excitation) 结构<sup>[19]</sup>。在多智能体强化学习中也存在注意力问题。每个智能体需要更加关注自己的任务,对和自己任务相关的态势要加强关注,对和自己不相关的态势要减少关注,甚至不关注。本文提出在智能体的策略网络中引入注意力机制,来加强智能体的决策能力。

在 MADDPG 算法中,每个智能体在与其他智能体进行信息交互时,是无差别的完全学习其他智能体的所有信息,这与实际情况并不相符。本文通过引入 Attention 机制,期望智能体在与其他智能体进行信息交互时,能够有选择性地关注能够让自己获得更大回报的那部分信息进行策略的学习。

为了使用注意力机制计算智能体的价值函数  $Q_i^w(o, a)$ , 需要对其进行改进。首先,定义 Critic 网络的输入是  $N$  个智能体的观测信息  $o = (o_1, o_2, \dots, o_N)$  和动作信息  $a = (a_1, a_2, \dots, a_N)$ 。

$$Q_i^w(o, a) = f_i(g_i(o_i, a_i), x_i) \quad (6)$$

式中:  $f_i$  是一个双层的 MLP 网络;  $g_i$  是一个单层 MLP 映射函数;  $x_i$  为其他智能体对当前智能体的贡献。

$$x_i = \sum_{j \neq i} \alpha_j v_j \quad (7)$$

式中:  $v_j$  为其他第  $j$  个智能体的价值函数;  $\alpha_j$  为注意力权值<sup>[20]</sup>。

本文在计算注意力权值时,权值  $\alpha_j$  是通过双线性映射,即查询-键值系统<sup>[21]</sup>来进行计算的,流程如下:每个智能体都可以将自己的观测动作对  $(o_i, a_i)$  传入到 MLP 网络  $g_i$  中得到编码  $e_i$ , 然后通过不同的网络进行处理得到“查询值”“键”“值”,对于智能体  $i$  来说,将自己的“查询值”以及其他智能体的“键”进行处理,并与其他智能体的“值”进行结合,得到其他智能体对该智能体的贡献加权之和  $x_i$ 。最后将  $x_i$  传递给另一个 MLP 网络  $f_i$  得到对应的动作价值函数  $Q_i(o_i, a_i)$ 。

## 3 仿真环境与多尺度奖励函数

### 3.1 仿真环境

#### 3.1.1 想定设计

在用于实验的仿真环境中, 作战双方各个作战单元的位置是随机的, 这样能够保证算法的鲁棒性。作战单元位置的随机性要满足实际情况, 即符合一定的限制条件。

作战双方分为红方和蓝方, 红蓝双方在一个长 500 km, 宽 500 km 的矩形海域内进行作战, 红方拥有 3 架 F-16AM MLU 型“战隼”战斗机, 蓝方拥有 2 架米格-29K 型舰载战斗机, 3 艘 PL-877M “基洛”级常规潜艇, 2 艘“戈尔什科夫”级护卫舰和 2 艘 Pr.956A 型“现代”级导弹驱逐舰。红方的作战任务是 3 架战斗机需要躲避蓝方战斗机, 相互配合, 尽快突袭蓝方的潜艇、护卫舰和驱逐舰; 蓝方的作战任务是要让本方战斗机追踪和打击红方战斗机。红方战斗机在作战场景的左上三角区内随机位置生成, 蓝方作战单元在作战场景的右下三角区内, 其中蓝方战斗机的位置在右下三角区内随机生成, 潜艇和护卫舰在固定区域内的随机位置生成, 驱逐舰则在右下角固定位置生成。作战双方的所有作战单元与本方其他单元之间的距离要大于一个阈值。一种可能的想定如图 3 所示。

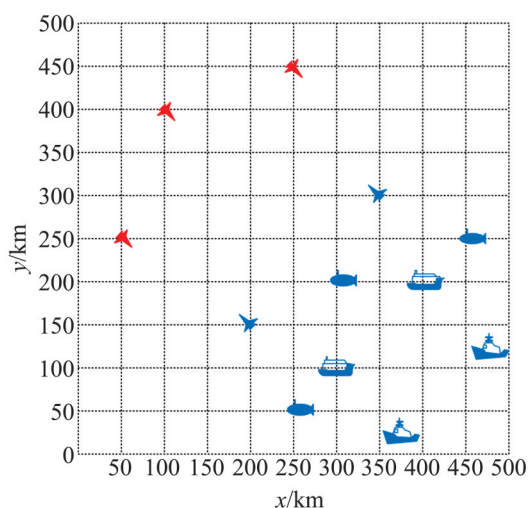


图 3 一种可能的想定初始态势

Fig. 3 A possible initial situation of scenario

#### 3.1.2 仿真环境设计

仿真环境使用 Python 语言 Gym 库提供的自定义仿真环境模型。Gym 定义了一套接口, 用于描述强化学习中的环境(env)这一概念, 同时在其官方库中, 包含了一些已实现的环境。Gym 库侧重于环境, 而非算法。该库既收录或实现了许多环境, 并且在其规则下, 也可以快速地构建环境。Gym 将环境抽象成一个类 env, 本文自定义的环境只需要继承该类并实现对应的接口即可。在 Gym 中定义了一些常用的空间类, Discrete 为其中之一, 它表示的是  $[0, n-1]$  的离散空间, 即  $0, 1, \dots, n-1$ 。Gym 的环境使用 reset 函数对环境进行重置, 通常在初始化 env 的时候就会 reset 一次; 使用 step 函数让推演进行一步, 需要在 render 中获取奖励, 智能体对环境的下一步观测以及本次推演是否结束标识; Gym 库还可以通过 render 函数进行环境的可视化, 可以实时看到推演效果。

在具体实现中, 首先在 PaddlePaddle 框架中搭建 Multi-agent Particle 环境, 以 BaseScenario 类为父类创建红蓝双方推演场景 Scenario 类。Scenario 类中包含整个推演环境的相关信息, 例如, 红蓝双方智能体个数、初始状态、随机属性, 以及奖励函数等。然后编写环境 MultiAgentEnv 类统一对各个 Scenario 类进行管理。MultiAgentEnv 类中包含了推演环境的初始化、状态空间和动作空间的设置、环境的每一步推演、重置环境、环境渲染等方法。MultiAgentEnv 类继承了 Gym 库的 env 类, 使用了空间类对状态空间和动作空间进行初始化, 并重新自定义 render 方法进行推演的显示。

### 3.2 多尺度奖励函数

本文设计的解耦模型中, 在中央控制器内有一个全局 Critic 网络和多个局部 Critic 网络, 每个局部 Critic 网络评价一个智能体的动作。智能体的决策网络通过 Critic 的评分来进行更新, 而 Critic 网络通过环境返回的奖励进行更新。解耦模型需



要设计全局奖励函数和局部奖励函数，全局奖励函数用来更新全局的 Critic 网络，局部奖励函数用来更新局部 Critic 网络。

在设计全局奖励函数时，需要其能够反映这一步推演给环境带来的整体影响，即让奖励函数能够综合所有智能体的动作所获得的奖励。由于一个局部的 Critic 网络对应一个智能体，所以在设计局部奖励函数时，要考虑到让奖励函数反映出该智能体动作给环境带来的影响，通过设计局部奖励函数，引导对应的智能体尽快完成任务。在设计奖励函数时，还要考虑到避免稀疏奖励的问题。

### 3.2.1 全局奖励函数

全局奖励函数需要体现所有智能体动作给环境带来的影响，红方的全局奖励函数分为 4 个部分，红方战斗机与蓝方潜艇、驱逐舰和护卫舰的距离作为正奖励，红方战斗机和蓝方战斗机的距离作为负奖励：

$$R_{FW} = \sum_{i=1}^3 \sum_{j=1}^3 \text{dis}(F_i, W_j) \quad (8)$$

$$R_{FD} = \sum_{i=1}^3 \sum_{j=1}^2 \text{dis}(F_i, D_j) \quad (9)$$

$$R_{FH} = \sum_{i=1}^3 \sum_{j=1}^2 \text{dis}(F_i, H_j) \quad (10)$$

$$R_{FS} = \sum_{i=1}^3 \sum_{j=1}^2 \text{dis}(F_i, S_j) \quad (11)$$

式中：dis()为距离函数； $F_i$ 为红方战斗机； $W_j$ 为蓝方潜艇； $D_j$ 为蓝方驱逐舰； $S_j$ 为蓝方战斗机； $H_j$ 为蓝方护卫舰。

红方的全局奖励函数为

$$R_{gr} = \alpha_1 R_{FS} - \alpha_2 R_{FW} - \alpha_3 R_{FD} - \alpha_4 R_{FH} \quad (12)$$

式中： $\alpha_1, \alpha_2, \alpha_3, \alpha_4$ 为奖励函数加权系数。

蓝方的全局奖励函数为

$$R_{gb} = -\beta R_{FS} \quad (13)$$

式中： $\beta$ 为奖励函数加权系数。

因为将作战单元之间的距离作为奖励函数的一部分，故每一步推演环境都会给出相应的奖励，从而避免了稀疏奖励的问题。

### 3.2.2 局部奖励函数

当前环境中需要设计奖励函数的作战单元为红方的 3 架战斗机和蓝方的 2 架战斗机。对于红方的 3 架战斗机，需要避免与己方其他战斗机碰撞，还要规避蓝方战斗机，同时还要对蓝方的潜艇和驱逐舰进行打击。根据任务需求，红方的战斗机与己方战斗机距离越大越好，与蓝方战斗机距离越大越好，与蓝方潜艇和驱逐舰距离越小越好。故红方战斗机的奖励函数设计如下。

(1) 定义红方战斗机碰撞惩罚回报：

$$R_{Cr} = \begin{cases} \sum_{i=1}^3 \text{dis}(F, F_i), & \text{dis}(F, F_i) < L \\ 0, & \text{dis}(F, F_i) > L \end{cases} \quad (14)$$

式中：dis()为距离函数； $L$ 为战斗机间的安全距离阈值； $F$ 为当前红方战斗机； $F_i$ 为红方战斗机。

(2) 定义红方战斗机和蓝方战斗机之间的距离惩罚回报：

$$R_{Sr} = \sum_{i=1}^2 \text{dis}(F, S_i) \quad (15)$$

式中： $S_i$ 为蓝方战斗机。

(3) 定义战斗机和蓝方潜艇、驱逐舰和护卫舰间的距离奖励回报：

$$R_D = \sum_{i=1}^3 \text{dis}(F, W_i) + \gamma_1 \sum_{i=1}^2 \text{dis}(F, D_i) + \gamma_2 \sum_{i=1}^2 \text{dis}(F, H_i) \quad (16)$$

式中： $W_i$ 为蓝方潜艇； $D_i$ 为蓝方驱逐舰； $H_i$ 为蓝方护卫舰； $\gamma_1$ 为打击驱逐舰的奖励系数； $\gamma_2$ 为打击护卫舰的奖励系数。

(4) 为了加强红方战斗机对蓝方作战单元的打击能力，定义加强打击引导奖励回报：

$$R_{F_1} = \begin{cases} R_1, & \text{dis}(F, W_i) < L \\ 0, & \text{dis}(F, W_i) > L \end{cases} \quad (17)$$

$$R_{F_2} = \begin{cases} R_2, & \text{dis}(F, D) < L \\ 0, & \text{dis}(F, D) > L \end{cases} \quad (18)$$

$$R_{F_3} = \begin{cases} R_3, & \text{dis}(F, H) < L \\ 0, & \text{dis}(F, H) > L \end{cases} \quad (19)$$

所以, 红方战斗机的局部奖励函数为

$$R_{lr} = \delta_1 R_{Cr} + \delta_2 R_{Sr} + \delta_3 R_{Dr} + \delta_4 (R_{F_1} + R_{F_2} + R_{F_3}) \quad (20)$$

式中:  $\delta_1, \delta_2, \delta_3, \delta_4$  为奖励函数加权系数。

蓝方战斗机的奖励函数设计如下。

(1) 定义蓝方战斗机碰撞惩罚回报:

$$R_{Cb} = \begin{cases} \sum_{i=1}^2 \text{dis}(S, S_i), & \text{dis}(S, S_i) < L \\ 0, & \text{dis}(S, S_i) > L \end{cases} \quad (21)$$

(2) 定义蓝方战斗机和红方战斗机之间的距离奖励回报:

$$R_{Sb} = \sum_{i=1}^3 \text{dis}(S, F_i) \quad (22)$$

所以, 蓝方战斗机的局部奖励函数为

$$R_{lb} = \varepsilon_1 R_{Cb} + \varepsilon_2 R_{Sb} \quad (23)$$

式中:  $\varepsilon_1, \varepsilon_2$  为奖励函数加权系数。

## 4 仿真实验

### 4.1 实验设置

实验场景中智能体的位置采取伪随机的机制, 红方的战斗机在环境右上部分随机位置生成, 蓝方战斗机在环境右下部分随机生成, 每个蓝方潜艇和护卫舰在右下部分固定区域随机生成, 保证各个作战单元的间距, 蓝方驱逐舰在右下角固定位置生成。

实验环境采用 Python 语言编写, 开发环境采用 PyCharm Community Edition 2020.3 版本和 Anaconda3 平台环境。深度学习框架采用飞桨 PaddlePaddle 2.2.0, 计算机配置为 CPU Intel(R) Core(TM) i7-9700 CPU @ 3.00 GHz, 显卡为 NVIDIA GeForce RTX 2070, 内存为 16 GB。实验设定的训练参数如表 1 所示。

### 4.2 实验展示

实验中的随机初始态势如图 4 所示。

红方战斗机会优先向蓝方驱逐舰进行攻击, 在经过蓝方潜艇和护卫舰附近时会转而对附近敌方单元进行攻击。该回合的最后一步态势如图 5 所示。

表 1 实验参数设置

Table 1 Experimental parameters setting

主要参数	量值
经验池容量 $M$	$10^5$
批样本数 Batchsize	1 024
折扣因子 $\gamma$	0.95
Critic 网络学习率 $\alpha_c$	0.01
Actor 网络学习率 $\alpha_a$	0.01
软更新率 $\tau$	0.01
最大回合数 Max Episode	5 000
每回合步数 Step Per Episode	25
安全距离阈值 $L/\text{km}$	2
演示时间步长/s	0.1

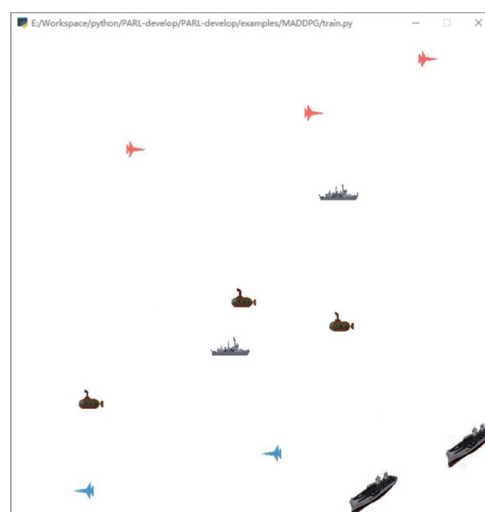


图 4 实验环境初始态势

Fig. 4 Initial situation of experimental environment

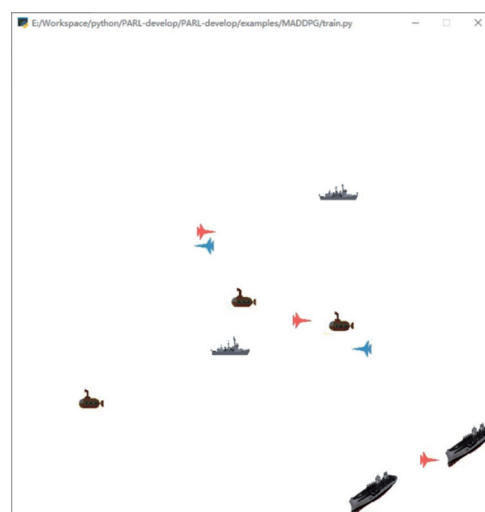


图 5 回合最后一步态势

Fig. 5 Situation of the last step of episode



红方战斗机之间会相互配合,由其中1架战斗机“引诱”蓝方战斗机,其余2架战斗机分别攻击敌方的潜艇、护卫舰和驱逐舰。

### 4.3 对比实验

为了验证算法的有效性,本文通过与MADDPG、QMIX<sup>[19]</sup>算法进行对比,分别记录不同算法的回合平均总奖励曲线,如图6所示。

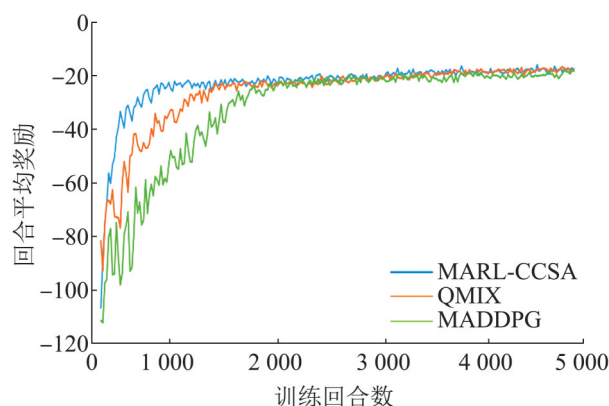


图6 回合平均总奖励

Fig. 6 Episode average total rewards

统计红方战斗机和蓝方战斗机的回合奖励,奖励曲线如图7和图8所示。

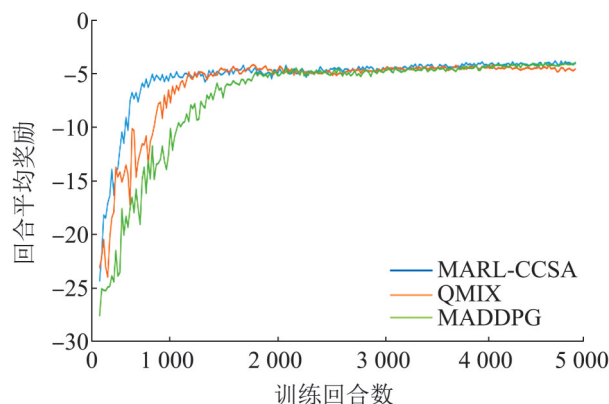


图7 红方战斗机回合平均奖励

Fig. 7 Episode average rewards of red fighter plane

实验结果表明,本文算法的红蓝方的所有战斗机的奖励值都取得了收敛的效果。传统的MADDPG算法收敛速度较慢,大约在1800~2000个回合之后达到收敛,收敛过程波动较大,且收敛不稳定。QMIX算法收敛速度相较于传统

MADDPG算法要快一些,大约在1300~1500个回合之后达到收敛,但是收敛曲线同样波动较大,收敛不稳定。本文算法相较于传统MADDPG算法和QMIX算法收敛速度最快,在大约500~700个回合之后就能够收敛,而且相较于其他2种算法,收敛更加稳定。

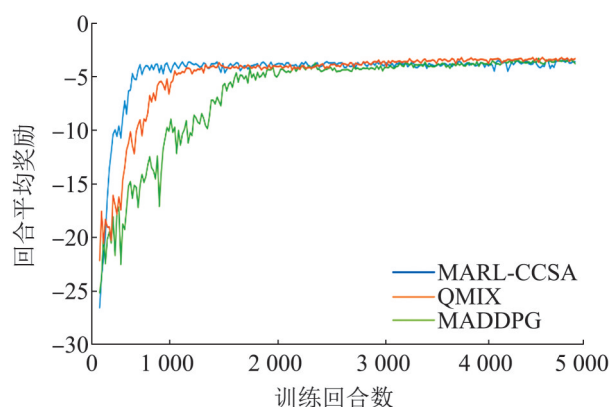


图8 蓝方战斗机回合平均奖励

Fig. 8 Episode average rewards of blue fighter plane

### 4.4 消融实验

为了验证本文提出的2种对MADDPG算法改进策略的有效性,设计消融实验,通过在原始的MADDPG算法上分别增加2种机制,比较不同模型的实验结果,实验结果如图9所示。

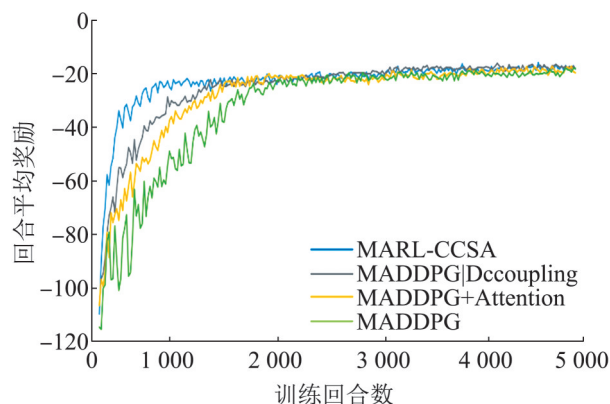


图9 不同模型实验结果

Fig. 9 Experimental results of different models

结果表明,在MADDPG算法的基础上加入注意力机制后,大约在1200~1400个回合之后模型达到收敛,而且模型收敛更加稳定;在MADDPG

算法上加入解耦机制,模型在训练1 000~1 200个回合之后能够达到收敛,大幅加快了算法的收敛。将解耦机制和注意力机制相结合,模型在训练500~700个回合之后达到收敛,说明注意力机制和解耦机制都能够提升模型的性能,并且在模型上将2个机制结合后,2种机制能够相互促进,取得更加优秀的性能。

## 5 结论

军事智能化是继机械化、信息化之后新一轮军事变革,是未来战场的必然发展趋势。对海战场环境进行作战仿真推演,从而把握未来智能化海战的战争形态和本质特征,才能更好地将智能技术应用到海战领域,获得对未来海战场决策优势的掌控。

本文针对智能海战场仿真问题,提出了一种基于改进的MADDPG算法模型,模型加入了解耦的优先经验回放机制和注意力Actor-Critic网络2种改进策略。同时,在解耦模型的基础上,提出一种多尺度的奖励函数,通过这种多尺度奖励函数加强对作战单元行为的引导。基于多尺度奖励函数构建了一个海战场仿真环境。在本文构建的仿真环境中进行对比实验,实验结果表明本文提出的算法模型在针对海战场想定的仿真推演有着较好的应用效果,相比于MADDPG和QMIX算法有着更好的表现,在消融实验中,实验结果表明本文提出的2种改进策略对算法模型的性能都有一定的提升。

下一步任务是利用强化学习将对敌情、我情、海情进行综合态势分析,快速生成军情分析报告,生成辅助决策方案;利用作战仿真技术对作战方案进行仿真推演,分析和评估作战方案;利用仿真推演分析和评估报告,对我方作战行动进行规划,作出合理的作战安排。

## 参考文献:

[1] 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. San Francisco: Margan Kaufmann, 2017.
- [2] Lillicrap T P, Hunt J J, Pritzel A, et al. Continuous Control with Deep Reinforcement Learning[C/OL]. International Conference on Learning Representations. 2016. [2022-06-11]. <https://arxiv.org/pdf/1509.02971>.
- [3] Rashid T, Samvelyan M, Schroeder C, et al. QMIX: Monotonic Value Function Factorisation for Deep Multi-agent Reinforcement Learning[C]//International Conference on Machine Learning. New York: PMLR, 2018: 4295-4304.
- [4] Watkins C J C H. Learning from Delayed Rewards[D]. London: King's College, 1989.
- [5] Rummery G A, Niranjan M. On-line Q-learning Using Connectionist Systems[M]. Cambridge, England: University of Cambridge, Department of Engineering, 1994.
- [6] 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. San Francisco: Margan Kaufmann, 2000: 1057-1063.
- [7] Mnih V, Kavukcuoglu K, Silver D, et al. Playing Atari with Deep Reinforcement Learning[J/OL]. [2022-06-11]. <https://arxiv.org/pdf/1312.5602>.
- [8] Barto A G, Sutton R S, Anderson C W. Neuronlike Adaptive Elements That Can Solve Difficult Learning Control Problems[J]. IEEE Transactions on Systems, Man, and Cybernetics(S0018-9472), 1983, 27(5): 834-846.
- [9] Hernandez-Leal P, Kartal B, Taylor M E. Is multiagent Deep Reinforcement Learning the Answer or the Question? A Brief Survey[J/OL]. [2022-06-11]. <https://arxiv.org/pdf/1810.05587>.
- [10] Tampuu A, Matiisen T, Kodelja D, et al. Multiagent Cooperation and Competition with Deep Reinforcement Learning[J]. Plos One(S1932-6203), 2017, 12(4): e0172395.
- [11] Gupta J K, Egorov M, Kochenderfer M. Cooperative Multi-agent Control Using Deep Reinforcement Learning [C]//International Conference on Autonomous Agents and Multiagent Systems. Cham: Springer, 2017: 66-83.
- [12] Foerster J N, Assael Y M, De Freitas N, et al. Learning to Communicate with Deep Multi-agent Reinforcement Learning[J]. [2022-06-11]. <https://arxiv.org/pdf/1605.06676>.
- [13] Sukhbaatar S, Fergus R. Learning Multi-agent Communication with Backpropagation[J]. Advances in Neural Information Processing Systems(S1049-5258), 2016, 29: 2244-2252.

- [14] Sunehag P, Lever G, Gruslys A, et al. Value-decomposition Networks for Cooperative Multi-agent Learning[J]. [2022-06-11] <https://arxiv.org/pdf/1706.05296>.
- [15] Foerster J, Nardelli N, Farquhar G, et al. Stabilising Experience Replay for Deep Multi-agent Reinforcement Learning[C]//International Conference on Machine learning. New York: PMLR, 2017: 1146-1155.
- [16] 符小卫, 王辉, 徐哲. 基于DE-MADDPG的多无人机协同追捕策略研究[J]. 航空学报, 2022, 43(5): 325311.  
Fu Xiaowei, Wang Hui, Xu Zhe. Cooperative Pursuit Strategy for Multi-UAVs Based on DE-MADDPG Algorithm[J]. Acta Aeronauticaet Astronautica Sinica, 2022, 43(5): 325311.
- [17] Schaul T, Quan J, Antonoglou I, et al. Prioritized Experience Replay[J]. [2022-06-11]. <https://arxiv.org/pdf/1511.05952>.
- [18] Vaswani A, Shazeer N, Parmar N, et al. Attention is All You Need[C]//Advances in Neural Information Processing Systems. San Francisco: Margan Kaufmann, 2017: 5998-6008.
- [19] Hu J, Shen L, Sun G. Squeeze-and-Excitation Networks [C]//IEEE Conference on Computer Vision and Pattern Recognition. Piscataway: IEEE, 2018: 7132-7141.
- [20] Iqbal S, Sha F. Actor-Attention-Critic for Multi-agent Reinforcement Learning[C]//International Conference on Machine Learning. New York: PMLR, 2019: 2961-2970.
- [21] Oh J, Chockalingam V, Lee H. Control of Memory, Active Perception, and Action in Minecraft[C]//International Conference on Machine Learning. New York: PMLR, 2016: 2790-2799.