DOI: 10.16725/j.1673-808X.202356
引用本文: 张琼, 赵岭忠, 翟仲毅. 基于 QMIX 的多智能体自适应探索方法[J]. 桂林电子科技大学学报, 2026, 46(3): 271-277.
Zhang Qiong, Zhao Lingzhong, Zhai Zhongyi. Adaptive exploration for multi-agent systems based on QMIX[J]. Journal of Guilin University of Electronic Technology, 2026, 46(3): 271-277.

基于 QMIX 的多智能体自适应探索方法

张 琼，赵岭忠，翟仲毅

（桂林电子科技大学 计算机与信息安全学院, 广西 桂林 541004）

摘 要： 针对目前多智能体强化学习方法中智能体通常采用贪心策略易导致对环境探索不充分而陷入次优解, 以及同一环境中所有智能体探索率相同而不足以适应复杂任务中的环境动态变化和其他智能体行为的问题, 提出了一种结合深度强化学习方法的多智能体自适应探索方法。该方法基于 QMIX 算法, 在 2 个时间层面上采用不同的方法调整智能体对环境的探索: 在每个时间步计算智能体的策略熵, 依据策略熵的大小按概率分布或状态动作值选择动作; 定义智能体在一段时间内的“表现”以及对环境的“探索程度”, 智能体参考整个团队在这段时间内的表现以及对环境的探索程度调整自己的探索率。在《星际争霸Ⅱ》场景中的 3s5z 与 5m_vs_6m 任务上进行了实验。实验结果表明, 相较于基线方法, 该方法在 2 种任务中都获得了更高的平均胜率, 显著提高了多智能体系统的学习性能。

关键词： 多智能体系统; 自适应探索; QMIX; 深度学习; 强化学习; 探索−利用平衡
中图分类号： TP301 **文献标志码：** A **文章编号：** 1673-808X(2026)03-0271-07

Adaptive exploration for multi-agent systems based on QMIX

ZHANG Qiong，ZHAO Lingzhong，ZHAI Zhongyi

（School of Computer and Information Security, Guilin University of Electronic Technology, Guilin 541004, China）

Abstract: To address insufficient exploration and premature convergence to suboptimal solutions in existing multi-agent reinforcement learning (MARL) methods, an adaptive exploration method based on deep reinforcement learning is proposed. Furthermore, using the same exploration rate for all agents limits their adaptability to dynamic environments and inter-agent interactions in complex tasks. Based on the QMIX framework, the proposed method adjusts agent exploration at two temporal scales. At each time step, the policy entropy is computed, and actions are selected according to a probability distribution derived from the state-action values. At a higher temporal scale, performance and exploration indicators are defined over a time window, and exploration rates are adaptively adjusted according to global team performance and exploration status. Experiments conducted on the 3s5z and 5m_vs_6m tasks in the StarCraft Ⅱ environment show that the proposed method consistently outperforms baseline methods and achieves higher average win rates in both tasks, significantly improving the learning performance of the multi-agent system.

Key words: multi-agent system; adaptive exploration; QMIX; deep learning; reinforcement learning; exploration-exploitation trade-off

近年来, 强化学习(reinforcement learning, 简称 RL)算法因其完成各种任务的能力而受到关注, 如游戏、涉及运动的复杂连续控制任务和工业应用[1-3]。在强化学习中, 智能体基于当前策略进行决策, 并收集经验数据来改进其行为。智能体在环境中的行为可分为“探索”行为和“利用”行为, 智能体尝试当前已知的最优动作之外的动作或者智能体选择目前已经探索出来的最优动作。当智能体做出“探索”

收稿日期: 2023-12-22
基金项目: 国家自然科学基金 (61862014)
通信作者: 赵岭忠(1977—), 男, 教授, 博士, 研究方向为软件模型检测、形式化技术。E-mail: zhaolingzhong163@163.com

行为时，它有可能会获得更多信息并提高自己的策略，但可能无法得到即时奖励。而当智能体做出“利用”行为时，它可以获得即时奖励，但可能导致次优决策[4]。

有研究者致力于促进单个智能体对环境的探索，Pathak 等[5] 提出了基于“好奇心”的探索，通过训练一个预测模型来预测下一个状态，然后计算预测状态与实际状态的偏差，偏差越大说明智能体对这个状态越不熟悉，并鼓励智能体到达偏差大的状态。Bellemare 等[6] 提出了基于“计数”的探索方法，对智能体访问过的状态进行计数，并鼓励智能体到达计数少的状态。Haarnoja 等[7] 提出了基于最大熵的探索策略方法，智能体在最大化预期收益的同时，还需要最大化策略熵。

在现实世界中，许多任务都需要多个智能体进行合作，诸如交通灯控制、传感器网络和多机器人编队等复杂问题[8-10]。在多智能体系统中，智能体不仅要与环境交互，还会受到环境中其他智能体的影响。有研究者致力于解决多智能体的联合探索问题，Mahajan 等[11] 介绍了分层控制的潜在空间，智能体根据潜在变量调整其行为，以执行协调探索。Roy 等[12] 探讨了如何通过策略正则化来促进多智能体深度强化学习中的协作探索。Wang 等[13] 研究了智能体的行为对其他智能体的影响，并鼓励智能体访问那些将改变其他智能体行为的“交互点”。Liu 等[14] 提出了 CMAE 算法，以确定值得探索的状态，并协调对这些状态的探索工作。

以上研究大多采用经典的探索方法，例如ε-贪心方法，该方法简单易用，但在学习策略时智能体易陷入次优解；此外，环境中各个智能体的探索率通常被设置为相同数值，而实际上，每个智能体在环境中的“经历”很可能不一样，有些智能体对环境的探索比较充分，而有些智能体对环境的探索不充分，因此这样设置不足以使智能体适应复杂任务中的环境动态变化和其他智能体的行为。

针对以上问题，提出了一种基于 QMIX 的多智能体自适应探索方法。从 2 个时间层面上用不同的方法调整智能体对环境的探索：1) 在每个时间步从策略层面上通过计算智能体策略熵的大小来选择按概率分布或是个体 Q 值采取动作。2) 每隔一段时间智能体通过参考团队对环境的整体“探索程度”以及“表现”调整自己的探索率。这样的设置既能避免由于智能体探索率线性衰减导致的对环境探索不充分的问题，又能够使得智能体有参考性地动态调整自己的探索率，从而实现多智能体系统中各个智能体自适应地探索环境，让智能体能够适应更加复杂多变的环境以及环境中其他智能体的行为，最终获得更高的回报。

1 相关知识

1.1 部分可观察马尔可夫决策过程

在多智能体系统中，智能体之间相互影响，相互作用，可以将多智能体强化学习过程建模为部分可观察马尔可夫决策过程 (decentralized partially observable Markov decision process，简称 Dec-POMDP)。通常，Dec-POMDP 可被定义为一个多元组$\langle N, S, O, A, T, R, \gamma\rangle$，其中：$N=\{1, 2, \cdots, n\}$是一个有限的智能体集合，表示环境中存在$n$个智能体；$S$ 表示所有智能体当前时刻的状态与环境信息；$O=\{O^1, O^2, \cdots, O^n\}$表示所有智能体的联合观察空间；$A=A^1A^2\cdots A^n$ 为所有智能体的联合动作空间；T 为状态转移函数，它定义了智能体在状态 S 下做出联合动作a后转移到状态S'的概率；R 为奖励函数，它将状态和联合动作的组合映射到单个标量奖励，在合作的多智能体系统中，这种奖励通常由所有智能体共享；$\gamma\in[0,1]$ 为折扣因子，它决定了智能体应该在多大程度上偏好获得即时回报而非长期收益。

1.2 QMIX 算法

QMIX[15] 算法是一种基于值分解的多智能体强化学习方法，是 Q-learning[16] 方法在多智能体系统中的扩展，其他基于值分解的方法还有 VDN、Qtran 和 Qatten[17-19]。演员−评论家方法也是一种常见的多智能体强化学习方法，比如 MADDPG、COMA、MAPPO、MAAC[20-23]。与基于值分解的方法不同的是，演员−评论家方法直接学习一个全局的评论家；而值分解方法不直接学习全局的 Q 函数，而是通过各个智能体的个体 Q 函数拟合一个全局的 Q 函数，智能体直接依据个体 Q 函数进行决策，个体 Q 函数用来生成个体 Q 值，表示智能体未来累积的预期回报值。通过最小化损失函数$L(\theta)$提升 Q 网络的准确性。

$$L(\theta)=\sum_{k=1}^{b}[(y_k^{\text{tot}}-Q(\tau,a,s;\theta))^2], \tag{1}$$

其中：b 为从缓冲区中采样的样本数；$Q(\tau,a,s;\theta)$为全局 Q 值；y^{tot}为目标值，

$$y^{\text{tot}}=r+\gamma\max_{a'}Q(\tau',a',s';\theta)。$$

1.3 最大熵策略方法与ε-贪心方法

最大熵策略[6] 方法需要同时最大化智能体的预期回报和策略熵，使得智能体的策略选择更多样化，避

免过于倾向某个特定动作。

ε-贪心方法是一种在强化学习中用于平衡探索和利用的方法。具体来说, 智能体以概率ε随机选择一个动作, 以概率$1-\varepsilon$选择当前状态下具有最大 Q 值的动作。

2　基于 QMIX 的多智能体自适应探索方法

在 QMIX 中, 智能体总是按照 Q 值的大小选择动作, 虽然有一定的探索率ε随机选择动作, 但ε往往很小, 因此在大多数时候智能体都是选择 Q 值最大的动作, 即使在各个动作 Q 值近似的情况下也会如此选择, 这样易导致智能体探索不充分, 从而陷入次优策略[11]。另外, 每个智能体在环境中的“经历”很可能不一样, 有些智能体对环境的探索足够充分, 而有些智能体对环境的探索不够充分, 因此对每个智能体设置一样的探索率不利于智能体学习, 且可能让智能体变“懒惰”。针对这些问题, 提出了基于 QMIX 的多智能体自适应探索方法(multi-agent adaptive exploration based on QMIX, 简称 MAAE-QMIX)。将智能体对环境的探索分为两部分, 一部分是通过策略去探索, 一部分是通过探索率去探索。在 2 个时间层面上对这两部分进行调整。在每个时间步计算智能体的策略熵, 依据策略熵的大小来决定智能体按动作的概率分布或根据 Q 值选择动作; 每过一段时间智能体通过参照环境中团队对环境的“探索程度”和“表现”调整自身的探索率。通过这 2 种调整方法最终达到促进智能体自适应探索的目的。

2.1　MAAE-QMIX 模型架构

MAAE-QMIX 模型是一个基于 QMIX 的多智能体自适应探索模型, 其主要思想是从 2 个时间层面上用不同的方法调整智能体对环境的探索, 实现智能体在环境中的自适应探索。

MAAE-QMIX 模型架构如图 1 所示, 由 MAAE-QMIX 模型的整体结构和单个智能体结构 2 个部分构成。图 1 中的混合网络旨在输出所有智能体的全局 Q 值$Q(\tau,a)$。而对于单个智能体i, 有一个代表个体值函数的网络, 旨在输出个体 Q 值$Q_i(\tau^i,a_t^i)$。

单个智能体内部结构如图 2 所示, 由 3 个部分构成: 1) 单个智能体的结构, 以智能体i在当前时间步接收到的个体观察o_t^i和在上一个时间步做出的动作a_{t-1}^i作为输入, 得到智能体的各个动作的 Q 值$Q_i(\tau^i,\cdot)$, 如图 2(b) 所示; 2) 策略调整模块, 输入智能体各个动作的 Q 值, 输出智能体选择的动作a_t^i以及该动作对应的 Q 值$Q_i(\tau^i,a_t^i)$, 如图 2(a) 所示; 3) 探索率调整模块, 将

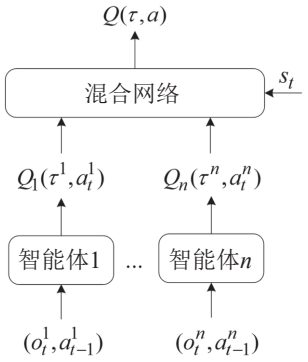
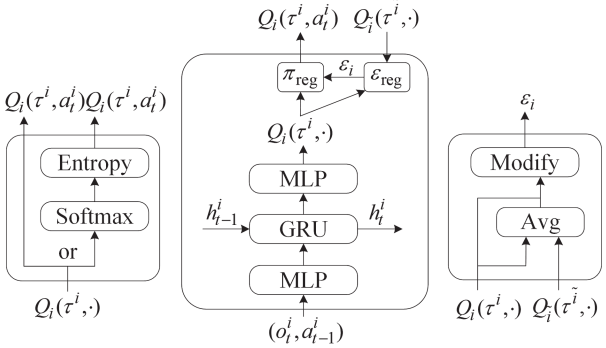


图 1　MAAE-QMIX 模型架构



(a) 策略调整模块　(b) 单个智能体结构　(c) 探索率调整模块

图 2　单个智能体内部结构

所有智能体的各个动作的 Q 值作为输入, 输出调整后的智能体的探索率ε_i, 如图 2(c) 所示。

2.2　策略调整模块

每个智能体都有自己的策略调整模块, 主要思想是在每个时间步从策略层面调整智能体的探索。具体来说, 在智能体各个动作的 Q 值差距较大的情况下鼓励智能体直接选择 Q 值大的动作; 而在 Q 值相似的情况下鼓励智能体从动作概率分布中采样动作, 以促进智能体的探索。该模块以智能体各个动作的 Q 值作为输入, 输出智能体选择的动作以及对应的 Q 值。

将智能体所有动作的 Q 值转换为对应的动作概率分布,

$$\pi_i(a|s)=\frac{\exp(Q_i(s,a)/\tau)}{\sum_{j=1}^{n}\exp(Q_j(s,a)/\tau)},\ i=1,2,\cdots,n,$$

其中: $\pi_i(a|s)$表示智能体在状态s下选择动作a的概率; $Q_i(s,a)$表示在状态s下选择动作a的 Q 值; τ表示温度参数, 用来控制概率分布的平滑程度。

策略熵用来判断各个动作的 Q 值的差距。

$$H_i(s)=-\sum_{a\in A}\pi_i(a|s)\log\pi_i(a|s),$$

当熵值大于$\beta(\ln|A|)$时，表示智能体各个动作的 Q 值差距小，此时鼓励智能体从概率分布中采样动作，促进智能体的探索；当熵值小于$\beta(\ln|A|)$时，表示智能体各个动作的 Q 值差距大，鼓励智能体选择 Q 值最大的动作，可表示为

$$a_t = \arg\max_a Q(s_t, a)。$$

$\ln|A|$为智能体动作空间大小的自然对数，当智能体各个动作的 Q 值相同时，熵值达到最大$\ln|A|$。$\beta \in (0,1]$，用来控制智能体在多大程度上倾向于用 Q 值选择动作。

通过策略调整模块，可从策略层面在智能体各个动作的 Q 值近似的情况下促进智能体对环境的探索。

2.3 探索率调整模块

探索率调整模块的主要思想是每隔一段时间从探索率层面调整智能体的探索。具体地，定义智能体在一段时间内的“表现”以及对环境的“探索程度”，智能体通过与团队的整体情况做比较来调整自己的探索率。该模块以当前智能体以及其他智能体各个动作的 Q 值作为输入，输出经过调整后智能体的探索率ε。

将智能体在一段时间的“表现”定义为智能体在这段时间内累积的预期回报值，

$$Q_{i,\text{tot}} = \sum_{t=1}^{T} Q_{i,t}(s_{i,t}, a_{i,t}),$$

则团队在这段时间内的整体表现为所有智能体在这段时间内表现的均值：

$$Q_{\text{avg}} = \frac{1}{n}\sum_{i=1}^{n} Q_{i,\text{tot}}。$$

定义智能体在一段时间内对环境的“探索程度”为智能体在这段时间内累积的策略熵值：

$$H_{i,\text{tot}} = \sum_{t=1}^{T}(-\sum_{a_t \in A}\pi(a_{i,t}|s_{i,t})\log\pi(a_{i,t}|s_{i,t})),$$

则团队在这段时间内对环境的整体探索程度为所有智能体在这段时间内对环境探索程度的均值：

$$H_{\text{avg}} = \frac{1}{n}\sum_{i=1}^{n} H_{i,\text{tot}}。$$

每个智能体以团队的整体表现以及对环境的探索程度为参考，调整自己的探索率。具体地，若智能体的表现比团队的整体表现好，则无需对智能体的探索率进行调整，让智能体按照目前的策略继续与环境交互；若智能体的表现比团队的整体表现差，且智能体对环境的探索程度高于团队对环境的整体探索程度，则需要适当减小智能体的探索率，$\varepsilon_i = \varepsilon_i - \Delta\varepsilon$；若智能体对环境的探索程度低于团队对环境的整体探索程度，则需要适当增加智能体探索率，$\varepsilon_i = \varepsilon_i + \Delta\varepsilon$。其中，$\Delta\varepsilon$为调整的量，且$\varepsilon_i \in [0.05, 0.15]$。

通过探索率调整模块，可以实现智能体参考团队的情况自适应地调整自身的探索率。

2.4 MAAE-QMIX 算法

模型的训练过程见算法 1。

算法 1 MAAE-QMIX 算法。

Initialize N individual Q networks Q_i and individual target Q networks Q'_i randomly
Initialize mixing network randomly
Initialize replay buffer D
cnt ← 1
for training episode = 1 to max_episodes do
  Each agent observe initial state s
  cnt ← cnt + 1
  for timestep t = 1 to episode_length do
  for agent i = 1 to N do
    Select action a_i through Policy
    Adjustment Module with probability ε_i
    if cnt == episode_num then
      Resize ε_i through Exploration Rate Adjustment Module
      cnt ← 1
    end if
  end for
  Execute actions $a = (a_1, a_2, \cdots, a_n)$ and observe reward r and next state s
  Add transition (s_t, a_t, r_t, s_{t+1}) to D
 end for
 if len(D) >= batch_size then
  Sample a random minibatch of transitions (s_t, a_t, r_t, s_{t+1}) from D
  Update Q network for each agent and mixing network by formula (1)
  Soft update Q' for each agent
  end if
 end for 。

算法 1 描述了 MAAE-QMIX 算法的整体训练过程，智能体通过策略调整模块与环境交互，并将与环境交互的样本数据(s_t, a_t, r_t, s_{t+1})存储在缓冲池 D 中，且每隔一段时间通过探索率调整模块调整ε，其中，

episode_num 是调整探索率的时间间隔。在存储了足够多的样本数据后, 每运行一个回合就从缓冲池中取出一批样本数据, 更新智能体的网络参数以及混合网络的参数。

3　实验与分析

为了验证和评估 MAAE-QMIX 算法的性能, 使用 epymarl 基准[24] 的场景, 方法用 PyTorch 实现, 所有实验均在 64 位 Windows 10 电脑上运行, 该电脑的配置为: Intel Core i5-8500 CPU @ 3.00 GHz, 8.00 GiB 内存, 开发环境为 PyCharm IDE。

3.1　实验环境描述

《星际争霸 Ⅱ》[25] 是一款即时战略游戏, 有一组完全合作的多智能体任务。每个任务都是一个特定的战斗场景, 包括一个智能体团队和一个敌方团队, 每个智能体控制一个单独的单位, 敌方团队由游戏内置脚本 AI 集中控制。每个团队的初始位置、数量和单位类型因场景而异。

观察与状态: 在每个时间步, 智能体接收在其视野范围内的局部观测, 包括视野范围内的地图信息以及存活的其他队友信息, 观察信息包括相对距离、x 坐标、y 坐标、生命值和单位类型。全局状态包含关于地图上所有单位的信息, 但仅在集中训练期间对智能体可用。

动作空间: 智能体在每个时间步可选择的离散动作有移动 [方向]、攻击 [敌人 ID]、停止和不操作。智能体只能对在其攻击距离内的敌人使用攻击 [敌人 ID] 动作。智能体的视野范围和攻击距离如图 3 所示, 其中蓝色圈为智能体的视野范围, 红色圈为智能体的攻击距离。

在每个回合开始, 由游戏内置脚本 AI 集中控制的敌方团队会指示其单位使用脚本策略攻击智能体团队。当任何一方团队的所有单位都死亡或达到预先指定的时间限制时(在这种情况下, 游戏被视为智能体团队的失败), 一个回合就结束了。智能体团队的目标是最大化胜率, 即赢得胜利的局数与总对战局数之比最大。

考虑《星际争霸 Ⅱ》中 2 个不同复杂程度的任务 3s5z 和 5m_vs_6m, 3s5z 指的是我方作战单位由 3 个追猎者(Stalker)和 5 个狂热者(Zealot)组成, 敌方作战单位亦由 3 个追猎者和 5 个狂热者组成, 如图 4(a) 所示。5m_vs_6m 指的是我方作战单位由 5 个陆战队员(Marine)组成, 敌方作战单位由 6 个陆战队员组成, 如图 4(b) 所示。

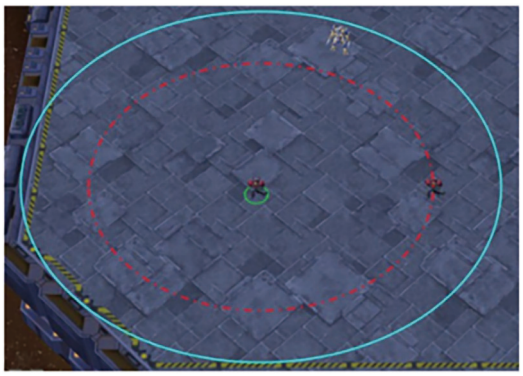


图 3　智能体的视野范围和攻击距离



(a) 3s5z



(b) 5m_vs_6m

图 4　所考虑的《星际争霸 Ⅱ》中的 2 个任务

3.2　实验结果分析

将 MAAE-QMIX 算法与 QMIX[15]、VDN[17] 算法进行比较。QMIX 与 VDN 是常见的基于值函数分解的方法, 都通过分解全局价值函数来解决多智能体协同决策问题, 不同的是, VDN 是将所有智能体的 Q 值相加得到全局 Q 值, 而 QMIX 通过对所有智能体的个体 Q 值函数进行加权混合得到全局 Q 值。

为了评估每种算法的性能, 共进行了 2 050 000 个时间步, 每 10 000 步暂停训练一次, 并在独立的测试环境中测试目标策略, 每个测试由 20 个测试回合组成, 用于计算平均测试奖励。实验参数设置: 缓冲池大小为 50, 学习率 α 为 0.000 5, 折扣系数 γ 为 0.99, 探索率调整量 $\Delta\varepsilon$ 为 0.001, β 为 0.95, episode_num 为

100。为了减少随机性的影响，分别用 2 个不同的随机种子来运行 MAAE-QMIX 算法和基线算法，且通过取平均值以获得更准确的结果，实验的评价指标为平均胜率。各算法在 2 个场景任务中的训练步数与平均胜率的关系如图 5 所示。

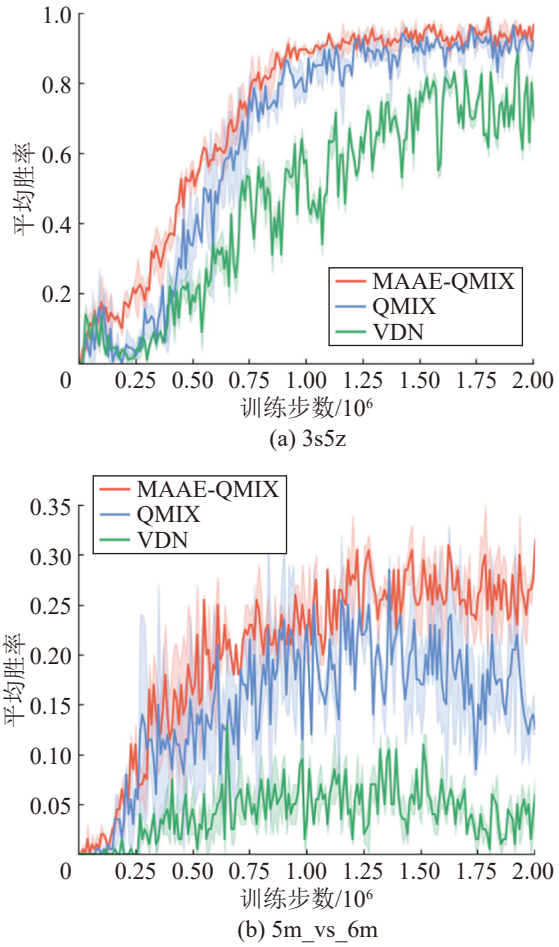


图 5　各算法在 2 个任务中的平均胜率

从图 5 可看出，3 种算法在 2 个任务中的表现都逐渐趋于平稳，但在相同时间内，相比于其他 2 种基线算法，MAAE-QMIX 算法能使智能体团队获得更高胜率。因此，基于 QMIX 的自适应探索模型能够促进智能体的探索，使智能体发现并学习更好的策略，并获得更高回合收益与平均胜率。

4　结束语

针对多智能体强化学习算法在探索方面存在的问题，提出了基于 QMIX 的多智能体自适应探索方法，该方法从 2 种时间层面上用不同的方法去调整智能体对环境的探索。在《星际争霸Ⅱ》的 3s5z 与 5m_vs_6m 场景上进行了实验，并与 QMIX 和 VDN 算法进行了比较。实验结果表明，该方法能有效提高智能体的学习性能，获得更高的回报。

参考文献：

[1] Perolat J, De VYLDER B, Hennes D, et al. Mastering the game of stratego with model-free multiagent reinforcement learning[J]. Science, 2022, 378(6623): 990-996.

[2] Kalashnikov D, Irpan A, Pastor P, et al. QT-opt: Scalable deep reinforcement learning for vision-based robotic mani-pulation[PP/OL]. arXiv (2018-11-28)[2023-05-10]. https://doi.org/10.48550/arXiv.1806.10293.

[3] Zhao Minghui, Guo Xian, Zhang Xuebo, et al. ASPW-DRL: assembly sequence planning for workpieces via a deep reinforcement learning approach[J]. Assembly Automation, 2019, 40(1): 65-75.

[4] Wiering M A. Explorations in efficient reinforcement learning[D]. Netherlands: Utrecht University, 1999: 71-72.

[5] Pathak D, Agrawal P, Efros A A, et al. Curiosity-driven exploration by self-supervised prediction[C]//2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops. New York: IEEE Press, 2017: 488-489.

[6] Bellemare M, Srinivasan S, Ostrovski G, et al. Unifying count-based exploration and intrinsic motivation [C]//Advances in Neural Information Processing Systems. New York, NY: Curran Associates, 2016: 1479-1487.

[7] Haarnoja T, Zhou A, Abbeel P, et al. Soft actor critic: off-policy maximum entropy deep reinforcement learning with a stochastic actor[J]. International Conference on Machine Learning. New York, NY: ACM Press, 2018: 1861-1870.

[8] Chen Chacha, Wei Hua, Xu Nan, et al. Toward a thousand lights: decentralized deep reinforcement learning for large-scale traffic signal control[J]. Proceedings of the AAAI Conference on Artificial Intelligence, 2020, 34(4): 3414-3421.

[9] Sahraoui M, Bilami A, Taleb-ahmed A. Schedule-based cooperative multi-agent reinforcement learning for multi-channel communication in wireless sensor networks[J]. Wireless Personal Communications, 2022, 122(4): 3445-3465.

[10] Alonso-mora J, Baker S, Rus D. Multi-robot formation control and object transport in dynamic environments via constrained optimization[J]. The International Journal of Robotics Research, 2017, 36(9): 1000-1021.

[11] Mahajan A, Rashid T, Samvelyan M, et al. Maven: multi-agent variational exploration[C]//Advances in Neural Information Processing Systems. New York, NY: Curran Associates, 2019: 7611-7622.

[12] Roy J, Barde P, Harvey F, et al. Promoting coordination through policy regularization in multi-agent deep reinforcement learning[J]. Advances in Neural Information

Processing Systems, 2020, 33(1323): 15774-15785.

[13] Wang Jianhong, Xu Wangkun, Gu Yunjie, et al. Multi-agent reinforcement learning for active voltage control on power distribution networks[J]. IEEE Transactions on Smart Grid, 2022, 13(6): 4873-4886.

[14] Liu I J, Jain U, Yeh R A, et al. Cooperative exploration for multi-agent deep reinforcement learning[C]//International Conference on Machine Learning. New York, NY: PMLR Press, 2021: 6826-6836.

[15] Rashid T, Samvelyan M, De WITT C S, et al. Monotonic value function factorisation for deep multi-agent reinforcement learning[J]. The Journal of Machine Learning Research, 2020, 21(1): 7234-7284.

[16] Mnih V, Kavukcuoglu K, Silver D, et al. Playing atari with deep reinforcement learning[PP/OL]. arXiv (2013-12-19)[2023-05-10]. https://doi.org/10.48550/arxiv.1312.5602.

[17] Sunehag P, Lever G, Gruslys A, et al. Value-decomposition networks for cooperative multi-agent learning[C]// Proceedings of the 17th International Conference on Autonomous Agents and Multi Agent Systems. Richland, SC: International Foundation for Autonomous Agentsand Multiagent Systems, 2018: 2085-2087.

[18] Son K, Kim D, Kang W J, et al. QTRAN: learning to factorize with transformation for cooperative multi-agent reinforcement learning[C]//International Conference on Machine Learning. New York, NY: PMLR Press, 2019: 5887-5896.

[19] Yang Yaodong, Hao Jianye, Liao Ben, et al. Qatten: a general framework for cooperative multiagent reinforcement learning[PP/OL]. arXiv(2020-06-09)[2023-05-10]. https://doi.org/10.48550/arXiv.2002.03939.

[20] Lowe R, Wu Yi, Tamar A, et al. Multi-agent actor-critic for mixed cooperative-competitive environments[C]//Proceedings of the 31st International Conference on Neural Information Processing Systems. New York, NY: Curran Associates, 2017: 6382-6393.

[21] Foerster J, Farquhar G, Afouras T, et al. Counterfactual multi-agent policy gradients[J]. Proceedings of the AAAI Conference on Artificial Intelligence, 2018, 32(1): 2974-2982.

[22] Yu C, Velu A, Vinitsky E, et al. The surprising effectiveness of PPO in cooperative multi-agent games[C]//Advances in Neural Information Processing Systems. Red Hook: Curran Associates, 2022: 24611-24624.

[23] Iqbal S, Sha F. Actor-attention-critic for multi-agent reinforcement learning[C]//International Conference on Machine Learning. New York, NY: PMLR Press, 2019: 2961-2970.

[24] Papoudakis G, Christianos F, Schäfer L, et al. Benchmarking multi-agent deep reinforcement learning algorithms in cooperative tasks[C]//Proceedings of the 1st International Workshop on Reinforcement Learning for Energy Management in Buildings and Cities. New York, NY: ACM Press, 2020: 15-19.

[25] 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.

编辑：张所滨