计算机科学
Computer Science
ISSN 1002-137X,CN 50-1075/TP

《计算机科学》网络首发论文

题目：基于 DQN 的多智能体深度强化学习运动规划方法
作者：史殿习，彭滢璇，杨焕焕，欧阳倩滢，张玉晖，郝锋
网络首发日期：2023-11-14
引用格式：史殿习，彭滢璇，杨焕焕，欧阳倩滢，张玉晖，郝锋．基于 DQN 的多智能体深度强化学习运动规划方法[J/OL]．计算机科学.
https://link.cnki.net/urlid/50.1075.tp.20231113.1335.022

网络首发时间：2023-11-14 15:00:46
网络首发地址：https://link.cnki.net/urlid/50.1075.tp.20231113.1335.022

基于 DQN 的多智能体深度强化学习运动规划方法

史殿习[1,3] 彭滢璇[2,3] 杨焕焕[2,3] 欧阳倩滢[1,3] 张玉晖[3] 郝锋[1]

1 智能博弈与决策实验室 北京 100091
2 国防科技大学计算机学院 长沙 410073
3 天津（滨海）人工智能创新中心 天津 300457
(dxshi@nudt.edu.cn)

摘 要 DQN 方法作为经典的基于价值的深度强化学习方法，在多智能体运动规划等领域得到了广泛应用。然而，DQN 方法存在一系列挑战，比如，DQN 会过高估计 Q 值、计算 Q 值较为复杂、神经网络没有历史记忆能力、使用ε-greedy 策略进行探索效率较低等等。针对这些问题，提出了一种基于 DQN 的多智能体深度强化学习运动规划方法，该方法可以帮助智能体学习到高效稳定的运动规划策略，无碰撞地到达目标点。首先，在 DQN 方法的基础上，提出了基于 Dueling 的 Q 值计算优化机制，将 Q 值的计算方式改进为计算状态值和优势函数值，并根据当前正在更新的 Q 值网络的参数选择最优动作，使得 Q 值的计算更加简单准确；其次，提出了基于 GRU 的记忆机制，引入了 GRU 模块，使得网络可以捕捉时序信息，具有处理智能体历史信息的能力；第三，提出了基于噪声的有效探索机制，通过引入参数化的噪声，改变了 DQN 中的探索方式，提高智能体的探索效率，使得多智能体系统达到探索-利用平衡状态。在 PyBullet 仿真平台的 6 种不同的仿真场景中进行了测试，实验结果表明，所提方法可以使多智能体团队进行高效协作，无碰撞地到达各自目标点，且策略训练过程稳定。

关键词： 多智能体系统；运动规划；深度强化学习；DQN 方法
中图法分类号 **TP391**

DQN-based Multi-Agent Motion Planning Method with Deep Reinforcement Learning

SHI Dian-Xi[1,3]，PENG Ying-xuan[2,3]，YANG Huan-huan[2,3]，OUYANG Qian-ying[1,3]，ZHANG Yu-hui[3] and HAO Feng[1]

1 Intelligent Game and Decision Lab（IGDL）, Beijing 100091, China
2 College of Computer，National University of Defense Technology，Changsha 410073， China
3 Tianjin Artificial Intelligence Innovation Center，Tianjin 300450，China

Abstract DQN as a classical value-based deep reinforcement learning method, has been widely used in the field of multi-agent motion planning. However, there are a series of challenges in DQN, such as, DQN can overestimate Q values, calculating Q values is more complicated, neural networks have no historical memory capability, using ε-greedy strategy for exploration is less efficient, etc. To address these problems, a DQN-based multi-agent deep reinforcement learning motion planning method is proposed, which can help the agents learn an efficient and stable motion planning strategy to reach the target points without collision. Firstly, based on the DQN method, an optimization mechanism for Q-value calculation based on Dueling is proposed, which improves the calculation of Q-value to calculate the state value and the advantage function value, and selects the optimal action based on the parameters of the Q-value network that is currently being updated, making the calculation of Q-value simpler and more accurate. Secondly, a memory mechanism based on GRU is proposed, and a GRU module is introduced, which enables the network to capture the temporal information and have the ability to process the historical information of the agents. Thirdly, an effective exploration mechanism based on noise is proposed, which changes the exploration mode in DQN by introducing parameterized noise, improves the exploration efficiency of the agents, and makes the multi-agent system reach the exploration-utilization equilibrium state. The experimental results have been tested in six different simulation scenarios in PyBullet simulation platform, and show that the proposed method can enable multi-agent teams to collaborate

基金项目： 科技部科技创新 2030—重大项目（No.2020AAA0104802）和自然科学基金（No.91948303）
This work was supported in part by Science and Technology Innovation 2030 Major Project under Grant (No.2020AAA0104802) and in part by the National Natural Science Foundation of China (No. 91948303).
通信作者：郝锋(no1haofeng@163.com)

efficiently and reach their respective target points without collision, and the strategy training process is more stable.

Keywords Multi-agent system, Motion planning, Deep reinforcement learning, DQN

1 引言

智能体是人工智能领域中一个重要概念，相较于单智能体，多智能体系统（Multi-Agent System，MAS）拥有高鲁棒性、高效性等许多不可替代的优势。运动规划问题[1]是多智能体系统的一个基础而核心的问题，如何进行高效的运动规划，使多个智能体协作完成复杂任务，始终是多智能体领域值得探索的问题。

传统的运动规划方法已经得到了广泛而深入的研究，例如常用的 A*算法[2]、蚁群优化方法（Ant Colony Optimization，ACO）[3]、人工势场方法（Artificial Potential Field，APF）[4]等。然而，这些方法无法处理复杂的高维信息，容易陷入局部最优，并不适用于复杂未知的环境，且存在计算复杂性高以及通用性能差等缺陷。

2013 年 Google DeepMind 的 Mnih 等人[5]首次提出了深度强化学习（Deep Reinforcement Learning，DRL）的概念，提出了深度 Q 网络（Deep Q Network，DQN），将强化学习（Reinforcement Learning，RL）和深度学习（Deep Learning，DL）技术相结合，在强化学习方法的基础上增加了深度卷积神经网络，实现了从高维数据到动作空间的端到端映射，该网络通过学习，仅基于图像输入即可进行 Atari2600 游戏。目前，DQN 方法被广泛应用于多智能体运动规划领域，突破了传统运动规划方法的瓶颈，通过智能体与环境不断交互产生的信息来进行策略学习，不需要任何先验知识；利用深度神经网络近似策略函数或者价值函数，大大提高了算法的泛化能力，对于运动规划问题具有很强的指导意义。

但是，多智能体运动规划场景具有很强的复杂性和多样性，DQN 方法本身具有一定的局限性，无法适应所有类型的多智能体场景。比如，DQN 方法由于根据 Q 目标网络的参数选择其中 Q 值最大的动作，通常会高估 Q 值，使得 Q 值的计算不够准确；DQN 方法将智能体的状态信息输入网络中，输出该状态下每个动作的动作价值，即一个与智能体动作空间维度相同的向量，而这需要对每个状态估计 N（动作空间大小）个 Q 值，计算较为复杂且很容易造成不稳定；DQN 方法的 Q 网络并没有历史记忆能力，通常需要将最近的 4 帧画面组成一个状态传入神经网络中进行学习，导致输入数据的维度和计算量的增加，从而导致 DQN 只适用于环境状态全部可观测的情况。但是，在现实世界中，存在很多部分可观察的环境，智能体无法观测到所有的环境信息，因此 DQN 难以解决该类问题。另一方面，在 DQN 方法中，使用ε-greedy 策略进行探索，智能体以一个较小的概率选择随机动作，探索效率较低。

综上，DQN 方法存在过高估计 Q 值、计算 Q 值较为复杂、神经网络没有历史记忆能力、探索效率较低等问题。为此，本文提出了一种基于 DQN 的多智能体深度强化学习运动规划方法，可以高效准确地计算出 Q 值、处理智能体的历史信息、使智能体高效探索状态空间，最终得出稳定的策略，使多智能体能够无碰撞地到达目标点，协作完成运动规划任务。本文的主要贡献包括：

（1）提出了一种基于 Dueling 的 Q 值计算优化机制，将 DQN 直接计算 Q 值的方式改进为计算状态值和优势函数值，并根据当前正在更新的 Q 值网络的参数选择最优动作，使得 Q 值的计算更加简单准确；

（2）提出了一种基于 GRU 的记忆机制，在 Q 网络中引入了 GRU 模块，使得网络可以捕捉时序信息，具有处理智能体历史信息的能力，从而可以使用智能体的单帧画面作为输入，不再需要将多帧画面堆叠起来作为输入；

（3）提出了一种基于噪声的有效探索机制，在神经网络的连接权重上添加了参数化的噪声，改变了 DQN 中的探索方式，使智能体以较高的效率进行探索，可使得多智能体系统达到探索-利用平衡状态，从而适合稀疏奖励等各类环境场景。

本文的其他部分组织如下：第 2 节描述了多智能体路径规划的相关方法和 DQN 方法的原理；第 3 节具体描述了本文所提出的基于 DQN 的多智能体深度强化学习运动规划方法，重点描述了该方法中的三个主要机制；第 4 节描述了仿真实验的设置以及实验结果；最后对本文工作进行了总结，并描述下一步工作。

2 相关工作

2.1 多智能体传统路径规划方法

路径规划是运动规划中的一个关键模块，多智能体路径规划（Multi-Agent Path Finding，MAPF）是指对多

个智能体的路径进行合理规划，避免智能体自身以及与环境之间的冲突[6]。根据获得的环境信息的差异，多智能体路径规划可分为全局路径规划和局部路径规划。全局路径规划即根据已知的环境地图选择一个完整的路径，常用的方法包括以 A*算法[2]为代表的基于搜索的方法、蚁群优化方法（ACO）[3]等。基于搜索的方法通过对状态空间进行搜索来寻找最优或次优的解，例如 A*[2]、CBS[7]等。A*算法是一种经典的启发式搜索方法，利用启发函数来指导智能体进行高效搜索，但随着智能体数量的增多，A*算法的搜索状态空间和搜索分支因子都会呈指数增长，从而导致 A*算法无法适用于较多智能体的路径规划问题。CBS 算法是一种中央规划算法，将多机规划问题分为两层，底层进行单机搜索，顶层利用约束树来遍历底层的规划路径处理冲突。ACO 算法的灵感来源于自然界中蚂蚁的觅食过程，蚂蚁在找到食物后返回巢穴的路途中，会留下一种信息素，为其他蚂蚁指引路径，单只蚂蚁按照概率随机选择归巢路径，较短的路径上单位时间内通过的蚂蚁数量更多，因此信息素的浓度更高，会吸引更多的蚂蚁选择该路径，在正反馈的作用下，最终整个蚁群会选择食物到蚁巢的最短路径，这也体现了群体智能的优越性。将该现象的思想应用于智能体的路径规划，起初所有蚂蚁的随机路径表示待求解问题的整个解空间，最终蚁群选择的最短路径即是该问题的最优解。全局路径规划方法依赖于已知的静态地图，因此难以在动态环境中使用。

局部路径规划方法包括人工势场法[4]（Artificial Potential Field，APF）和动态窗口法[8]（Dynamic Window Approach，DWA）等。APF 方法假设智能体在一种虚拟力场下运动，包括引力场和斥力场，目标点对智能体产生引力，而障碍物对智能体产生斥力。DWA 方法包括速度空间的计算和速度空间的评价两个部分，首先，在速度空间内对智能体的线速度和角速度进行采样，根据智能体的运动学模型预测其下一个时间段的运动轨迹；其次，对运动轨迹进行评分，从而获得较为安全平滑的最优局部路径。APF 和 DWA 等局部路径规划方法的特点是可以有效处理环境的动态变化，重新规划局部路径。

传统路径规划方法遇到的瓶颈包括计算复杂性高、实时性能差、不适用于复杂未知环境等，这在很大程度上限制了多智能体系统的发展。多智能体路径规划领域还有许多待解决的挑战和问题，如动态环境、异构智能体、不完全信息等。

2.2 多智能体深度强化学习路径规划方法

强化学习技术具有较好的决策能力，但缺乏对于高维信息的感知能力，深度学习恰恰可以弥补这方面的不足。深度强化学习技术将强化学习与深度学习相结合[9]，充分发挥二者优势，首先，智能体与环境交互获得高维数据，利用深度学习方法进行感知，以得到环境状态的特征表示；其次，利用强化学习方法进行决策，将当前状态端到端地映射为智能体的动作。深度强化学习方法将多智能体系统运动规划描述为一个马尔可夫决策过程（Markov Decision Process，MDP）[10]，以传感器的观察作为状态，通过与环境的交互，深度强化学习方法可以找到引导智能体到达目标位置的最优策略。深度强化学习方法使得多智能体系统得以自主学习运动策略，规划合理的运动路径，具有无地图、学习能力强、对传感器精度依赖性低、计算复杂度低、通用性强等优点[11]，能够有效应对传统运动规划方法遇到的瓶颈问题，因此，基于深度强化学习的运动规划问题成为多智能体协同领域研究的关键问题之一[12]。

Di Wang 等人[13]的团队关注依赖视觉的多智能体运动规划问题，他们基于 DQN 方法，提出了端到端的 MRCDRL 方法，以从机器人的第一人称视角收集到的图像信息作为输入，经过神经网络结构后，输出机器人的动作，可以有效应对任务分配和路径规划问题，但是该方法仅考虑简单的二维场景，无法适用于复杂的三维场景。2021 年，该团队针对三维环境，提出了一种多机器人协调算法[14]，该方法采用 TFDueling 神经网络结构，首先，顶视图流和第一人称视图流分别使用卷积神经网络来提取顶视图和第一人称视图的特征；其次，Dueling 流将两个特性相结合并输出相应的机器人动作。该方法证明了 Dueling 结构对于网络架构有积极的贡献作用，但该方法采用ε-greedy 贪婪策略，机器人以较小的概率进行动作探索，且仍然以 DQN 连续 4 帧画面的方式进行输入，没有考虑神经网络的历史信息处理问题。Peter Sunehag 等人[15]基于 DQN 方法，研究了具有单一联合奖励信号的协同多智能体强化学习问题，这类学习问题通常具有很大的组合动作和观察空间。由于环境的部分可观察性，存在虚假奖励和“懒惰智能体”的问题，作者通过使用一种新的值分解网络结构训练单个智能体来解决这些问题，该网络结构学习将团队值函数分解为基于智能体的值函数。实验结果表明，当与权重共享、角色信息和信息通道相结合时，价值分解可以产生更好的结果。

2.3 DQN 方法

DQN[16]是一种基于 Q-Learing 的强化学习方法，同时是一种典型的基于价值的深度强化学习方法，该方法将神经网络和 Q-learning 相融合，在多智能体强化学习领域取得了巨大的成功。Q-Learing 算法维护一个 Q-table，使用表格存储每个状态s下采取动作a获得的奖励，即状态-价值函数 $Q(s,a)$，但是，这种算法存在很大的局限性，在现实中很多情况下，强化学习任务所面临的状态空间是连续的，存在无穷多个状态，在这种情况下，无法再使用表格的方式存储价值函数。为了解决这个问题，可以用一个函数 $Q(s,a;w)$来近似动作-价值 $Q(s,a)$，称为价值函数近似（Value Function Approximation），用神经网络来生成该函数 $Q(s,a;w)$，称为 Q 网络，w 是神经网络训练的参数。

式(1)是 DQN 算法的目标函数，其中θ^-表示一个固定且独立的目标网络的参数。DQN 由于根据 Q 目标网络的参数选择其中 Q 值最大的动作，通常会高估 Q 值，针对这一问题，DDQN[17]方法根据当前正在更新的 Q 值网络的参数选择最优动作（如式(2)所示），在一定程度上降低了对 Q 值的高估，使得 Q 值更加接近真实值。

$$Y_t^{DQN} = r_i^t + \gamma max_{a_i^{t+1}} Q\left(s_i^{t+1}, a_i^{t+1}; \theta^-\right) \tag{1}$$

$$Y_t^{DoubleDQN} = r_i^t + \gamma Q\left(s_i^{t+1}, \underset{a_i^{t+1}}{argmax} Q\left(s_i^{t+1}, a_i^{t+1}; \theta^t\right); \theta^-\right) \tag{2}$$

3 方法

针对 DQN 方法存在的问题，为有效提升多智能体系统的运动规划性能，本文针对性地提出了一种适用于多智能体运动规划的深度强化学习方法 SDQN（Super Deep Q Network）。SDQN 采取集中训练分布执行范式，在训练过程中，所有智能体之间共享策略，训练完成后，每个智能体分散独立执行各自的策略。SDQN 方法总体框架如图 1 所示，主要由多智能体意图预测机制、基于 GRU 的记忆机制、基于噪声的有效探索机制和基于 Dueling 的 Q 值计算优化机制等几个部分构成。其中，多智能体意图预测机制采用我们前期工作中的意图预测方法 MA_2IMP[18]，将意图概念引入到多智能体运动规划当中，将智能体的视觉图像和历史地图相结合以预测智能体的意图，使智能体可以对其他智能体的动作作出预判，从而有效支持智能体之间的协作（有关多智能体意图预测机制详细内容见文献[18]）。SDQN 方法的核心是策略训练机制，以 DDQN 方法为基础，基于 GRU 的记忆机制引入了 GRU 模块，使得网络可以捕捉时序信息，具有处理历史信息的能力；基于噪声的有效探索机制主要改变了 DQN 中的探索方式，引入了参数化的噪声，提高智能体的探索效率，使得多智能体系统达到探索-利用平衡状态；基于 Dueling 的 Q 值计算优化机制将 DDQN 中的 Q 值计算方式改进为计算状态值和优势函数值，使得 Q 值的计算更加简单准确。

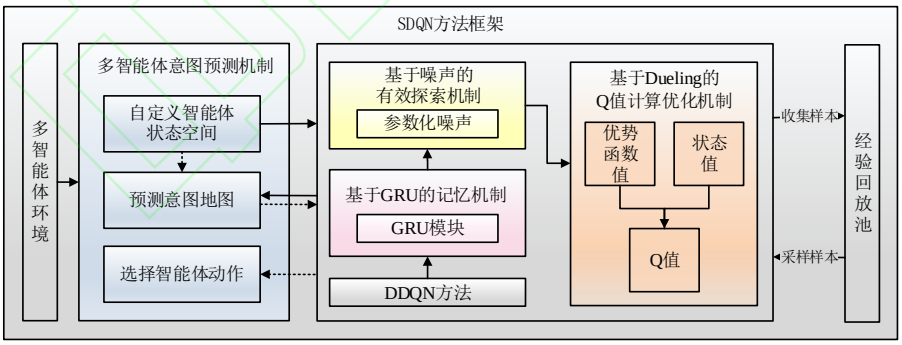


图 1 SDQN 方法整体框架
Fig.1 Overall framework of SDQN

SDQN 方法的工作原理如下：首先，多智能体意图预测机制使用智能体获得的视觉数据和历史地图进行状态表示，将状态表示输入神经网络，利用基于 GRU 的记忆机制和基于噪声的有效探索机制进行策略训练，得到预测意图地图；其次，将预测意图地图和视觉数据、历史地图一起，组成新的状态表示，再次输入网络，利用基于

GRU 的记忆机制、基于噪声的有效探索机制和基于 Dueling 的 Q 值计算优化机制等 3 种机制进行策略训练，最终得到智能体的动作。

基于 GRU 的记忆机制、基于噪声的有效探索机制和基于 Dueling 的 Q 值计算优化机制等 3 种机制均可以通过优化神经网络结构来进行实现，SDQN 方法的网络结构如图 2 所示，包括注意力模块、GRU 模块、NoisyLinear 模块、卷积模块和 Dueling 模块。在 SDQN 网络中，首先，在 ResNet18 第一层之后添加通道注意力模块，从而使网络关注更有意义的智能体感知信息，在通道注意力模块之后，进一步引入空间注意力模块来关注输入图像中哪些区域的特征更有意义；其次，在 ResNet18 的 4 个 layer 层之后增加 GRU 模块，用于获取上一层的隐藏状态；第三，在 GRU 模块之后增加 NoisyLinear 模块，对每个神经元添加高斯分解噪声，从而将权重和偏置的噪声参数加入学习网络；第四，在卷积模块，我们添加了三个卷积层，并在每个卷积层之后应用 BatchNorm，以使每一层神经网络的输入保持相同的分布；最后，在卷积模块之后，不直接输出 Q 值，而是添加一个 Dueling 模块，将 Q 值的计算分解为状态值和优势函数的计算，再输出最终的 Q 值和智能体的动作选择。

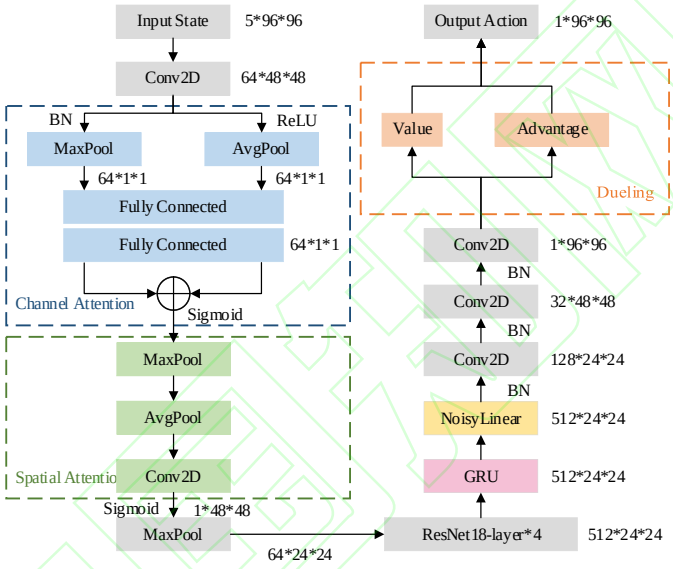


图 2 SDQN 网络结构图

Fig.2 SDQN network architecture

3.1 基于 Dueling 的 Q 值计算优化机制

虽然 DDQN 方法通过改进目标函数的计算方式来应对 DQN 方法中过高估计 Q 值的问题，显著提升了 DQN 的训练效果，但是以 DQN 为代表的基于价值的方法依然存在一系列挑战，我们主要针对策略训练算法进行改进优化，以提升多智能体系统的策略训练性能。

在原始 DQN 方法中，将智能体的状态信息输入网络中，输出该状态下每个动作的动作价值，即一个与智能体动作空间维度相同的向量，这需要对每个状态估计N（动作空间大小）个 Q 值，很容易造成不稳定，因此本文对网络的输出结构进行改进优化，提升其性能。

在多智能体运动规划场景中，当智能体前方没有障碍物时，智能体左右移动对 Q 值没有影响，但是状态值对 Q 值有较大影响；当前方存在障碍物时，智能体的动作选择至关重要，如果不采取相应动作，智能体很有可能会发生碰撞，此时动作和状态均对 Q 值产生影响。因此，我们可以发现，在许多情况下，智能体的当前状态对于 Q 值起决定性作用。因此，本文对智能体的状态价值进行计算，再与动作价值相加，得到最终的 Q 值。DQN 的动作价值函数如式(3)所示：

$$Q_\pi(s_t,a_t)=E[U_t \mid S_t=s_t,A_t=a_t] \tag{3}$$

其中，U_t表示折扣回报，如式(4)所示：

$$U_t=R_t+\gamma R_{t+1}+\gamma^2 R_{t+1}+\gamma^3 R_{t+3}+\cdots \tag{4}$$

即从 t 时刻开始，未来所有时刻的奖励加权求总和。式(3)中的$Q_\pi(s_t,a_t)$即折扣回报的条件期望，依赖于当前 t 时刻的状态s_t、动作a_t以及策略π。

最优的动作价值函数即在策略π下，对$Q_\pi(s_t,a_t)$求最大值，如式(5)所示：

$$Q^*(s,a) = max_\pi Q_\pi(s,a) \tag{5}$$

状态价值函数可以表示为式(6)：

$$V_\pi(s_t) = E[Q_\pi \mid s_t, A] \tag{6}$$

其中，$V_\pi(s_t)$表示$Q_\pi(s_t,a_t)$的条件期望，与动作a_t无关，依赖于当前 t 时刻的状态s_t和策略π。

最优的状态价值函数即在策略π下，对$V_\pi(s_t)$求最大值，如式(7)所示：

$$V^*(s) = max_\pi V_\pi(s) \tag{7}$$

当动作价值函数最优时，状态价值函数也到达最优，因此式(7)也可以表示为式(8)：

$$V^*(s) = max_a Q^*(s,a) \tag{8}$$

由此可以推导出，最优的优势函数如式(9)所示：

$$A^*(s,a) = Q^*(s,a) - V^*(s) \tag{9}$$

表示以$V^*(s)$作为基线，动作 a 相较于基线的优势，优势越大，说明动作 a 越好。我们对式(9)进行推导，首先等式两边同时对动作 a 取最大值，得到式(10)：

$$max_a A^*(s,a) = max_a Q^*(s,a) - V^*(s) \tag{10}$$

根据式(8)可得式(11)：

$$max_a A^*(s,a) = 0 \tag{11}$$

对式(9)进行变换，可得式(12)：

$$Q^*(s,a) = V^*(s) + A^*(s,a) \tag{12}$$

但是，在式(12)中，$V^*(s)$和$A^*(s,a)$存在多对解，即便确定了$Q^*(s,a)$的值，也无法得出唯一的$V^*(s)$和$A^*(s,a)$，这样就导致，如果神经网络 V 和 A 进行方向相反、幅度相同的波动时，其输出结果相同，两个网络不停波动，必然会导致训练效果不佳。为了避免这一不确定性，我们引入式(11)的最大化项，可得到最优 Q 值的求解公式如式(13)所示：

$$Q^*(s,a) = V^*(s) + A^*(s,a) - max_a A^*(s,a) \tag{13}$$

最大化和均值是理论研究中常用的两种表示形式，在实际的训练过程中我们发现，均值的效果比最大化效果更好，因此，我们将式(13)中的最大化项更改为均值项。

在多智能体运动规划任务中，算法的当前输入状态是s_i^t，其中，t代表第t个时间步，i代表机器人的编号，$i \in [1,n]$，n是机器人的数量。每个时间步的经验序列$e_i^t = (s_i^t, a_i^t, r_i^t, s_i^{t+1})$被添加到经验池$\mathcal{D} = \{e_i^1, \dots, e_i^t\}$中，其中，$a_i^t$代表机器人$i$在第$t$个时间步执行的动作。我们可以将 Q 值的求解[19]表示为式(14)：

$$Q(s_i^{t+1}, a_i^{t+1}; \theta^t, \alpha^t, \beta^t) = V(s_i^{t+1}; \theta^t, \beta^t) + \left(A(s_i^{t+1}, a_i^{t+1}; \theta^t, \alpha^t) - \frac{1}{|\mathcal{A}|}\sum A(s_i^{t+1}, a_i^{t+1}; \theta^t, \alpha^t)\right) \tag{14}$$

其中，$V(s_i^{t+1}; \theta^t, \beta^t)$表示状态值，只与智能体的状态相关，与动作无关；$A(s_i^{t+1}, a_i^{t+1}; \theta^t, \alpha^t)$是优势函数，用于衡量每个动作相对于其他动作有多好；$\frac{1}{|\mathcal{A}|}\sum A(s_i^{t+1}, a_i^{t+1}; \theta^t, \alpha^t)$是优势函数的均值；$\theta^t$是卷积层参数；$\alpha^t$是优势函数的网络参数；$\beta^t$是状态值函数的网络参数。

DQN 利用神经网络$Q(s,a;\omega)$来近似最优的动作价值函数$Q^*(s,a)$，从而得到智能体的最优策略；本文利用神经网络$A(s,a;\omega^A)$来近似最优优势函数$A^*(s,a)$，利用神经网络$V(s;\omega^V)$来近似最优状态价值函数$V^*(s)$，其中，ω^A和ω^V均表示网络参数。图 2 的 Dueling 部分展示了 Q 值分解的网络架构。基于 Dueling 的 Q 值计算优化机制，将 Q 值函数分解为状态值函数和优势函数两个部分，这样可以更好地估计不同动作对于状态的贡献，提高学习效率。

3.2 基于 GRU 的记忆机制

在一些动态环境中，仅凭一帧画面无法判断物体的运动方向和速度等信息，因此 DQN 算法通常需要将最近的 4 帧画面组成一个状态传入神经网络中进行学习，从而增加输入数据的维度和计算量，并且可能导致时间上的延迟。可见，DQN 算法存在着两点局限性，一是存储经验数据的内存有限，二是需要完整的观测信息，从而导致 DQN 只适用于环境状态全部可观测的情况；但是在现实世界中，存在很多部分可观察的环境，智能体无法观测到所有的环境信息，因此 DQN 难以解决这类问题。

如果 DQN 具有处理历史信息的能力，在当前时刻可以根据历史信息进行决策，则不再需要足够的内存和完整的观测信息，也不再需要同时输入 4 帧画面进行学习，上述问题便迎刃而解。基于这一思路，Matthew Hausknecht 和 Peter Stone 提出了 DRQN 算法[20]，其本质是将长短期记忆网络（Long Short-Term Memory，LSTM）与 DQN 相结合，LSTM 长短期记忆网络是一种特殊的循环神经网络（Recurrent Neural Networks，RNN）类型。RNN 的设计灵感来源于大脑皮层的循环反响回路，该神经网络的关键在于递归结构，它具有记忆历史信息的能力，专门用于处理语言、语音、时间序列等序列数据[21]，RNN 的结构如图 3 所示。RNN 和全连接神经网络的本质差异在于“输入是带有反馈信息的”，RNN 的核心是一个有向图，有向图中链式相连的单元是循环单元，每个循环单元的结构和功能类似，假设当前时刻为t，当前时刻的隐藏状态h_t由当前时刻的输入信息X_t和上一时刻的隐藏状态h_{t-1}共同决定，每个循环单元的输出由当前时刻的隐藏状态h_t决定。可见，

RNN 的隐藏状态利用了上一步的历史反馈信息，这更符合大脑的决策原理，大脑利用当前时刻的感官信息和之前的想法一起进行决策。

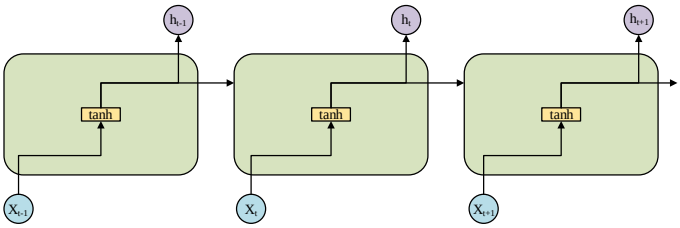


图 3 RNN 结构图

Fig.3 RNN architecture

在实际应用当中，RNN 也存在一定的缺陷，如果处理的序列数据过长，RNN 需要记忆大量历史信息，此时，很容易出现一系列问题，比如遗忘较早的信息（梯度消失）或者系统崩溃（梯度爆炸）等，即所谓的长期依赖（long-term dependencies）问题。在长序列数据处理中，并不是所有的历史信息都是有用的，RNN 记忆无用的信息会造成资源的浪费，因此，如果能够让 RNN 有选择地记忆和遗忘一部分信息，则可以很好地应对上述问题，这也是 LSTM 的基本思想。

LSTM 针对 RNN 的长期依赖问题，设计了门控算法，通过门控单元控制内部信息的积累，设计了 3 个“门控”，分别是遗忘门、输入门和输出门[22]，3 个门控对 LSTM 单元的内部状态建立了自循环，既能较好地处理长期依赖问题，又能选择性地删除信息，防止记忆过载，其结构如图 4 所示。LSTM 第一步是通过“遗忘门”，从上个时刻的状态中丢弃一些信息；随后通过“输入门”，决定将在状态中存储哪些新信息；最后通过“输出门”，根据上一时刻的输出和当前时刻的输入，输出最终信息。LSTM 具有时间循环结构，可以很好地刻画具有时空关联的序列数据，类似于一种基于神经网络的自回归模型。

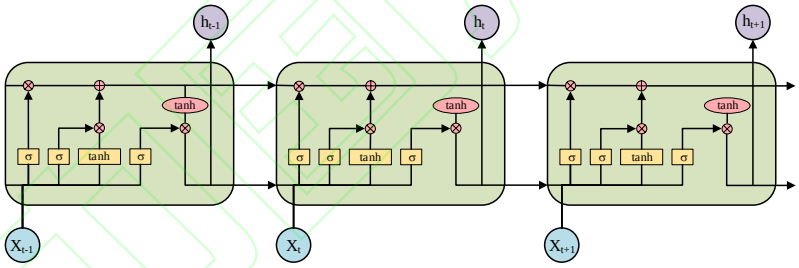


图 4 LSTM 结构图

Fig.4 LSTM architecture

我们认为，在多智能体运动规划问题中同样可以借鉴 DRQN 的思想，为此，我们将 RNN 与 DQN 相结合，以增强智能体的记忆能力和推理能力，使智能体可以从历史观察中提取有用的信息，并在信息丢失或不确定的情况下选择合理的动作。

LSTM 模型结构较为复杂，会引入较多的网络参数，在一定程度上影响学习效率，为了简化网络结构，提升网络的性能，我们在神经网络架构的 ResNet18 的 4 个 layer 层之后添加一个 GRU 模块，其结构如图 5 所示。GRU 与 LSTM 相比，构造更加简单，降低了对学习效率的影响。GRU 使用重置门和更新门，重置门作用于前面的隐藏状态，决定需要遗忘多少历史信息；更新门类似于 LSTM 中的遗忘门和输入门相结合，作用于当前时刻和上一时刻的隐藏单元，决定需要向下传递多少有用信息。

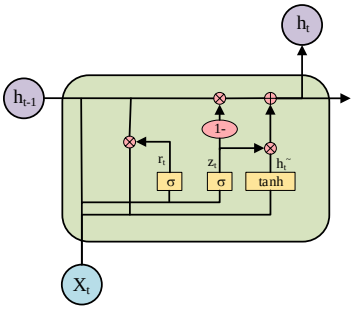


图 5　GRU 结构图
Fig.5　GRU architecture

由于 GRU 比 LSTM 减少一个门，从而减少了矩阵乘法，因此在训练期间需要更新的权重和参数数量更少，尤其是在在训练数据很大的情况下，GRU 不仅能够有效地节省训练时间，而且可以减少输入数据的维度。与 DQN 需要将多帧画面堆叠起来作为输入不同，添加 GRU 模块后，可以使用智能体的单帧画面作为输入，利用 RNN 来捕捉时序信息，并且能够达到比 DQN 更好的性能。

3.3　基于噪声的有效探索机制

强化学习中一个非常基础而关键的问题就是探索和利用问题[23]。探索是指智能体进行之前从未进行过的新行为，以期望得到更高的回报；利用是指智能体从已经进行过的动作中选取能够产生最大回报的动作。如何寻找探索和利用之间的平衡，是强化学习面临的一个困境问题[24]，尤其当环境包含大的状态空间、欺骗性或稀疏奖励时，有效探索显得更加重要。

目前，常见的探索方法主要有 epsilon 贪婪策略和熵正则化策略等。其中，DQN 方法使用 epsilon 贪婪策略进行智能体的动作探索，在该策略中，智能体以一个较小的概率选择随机动作；熵正则化策略则是在标准奖励的基础上，加上一个熵正则化项来奖励随机化的策略。这两种方法都是在智能体的动作上添加噪声，导致探索的效率较低。本文提出的 SDQN 方法则是在神经网络的连接权重上添加参数化的噪声，以提高强化学习算法的探索效率。

在本文设计的神经网络架构中，原本的全连接层需要学习的只有权重w，我们将噪声ε^w和ε^b作为参数加入到全连接层的连接权重上，这样一来，全连接层既要学习w的均值μ，又要学习w的方差σ，且均值和方差都作为网络参数进行学习，不需要人工调整，加入参数化噪声的示意图如图 6 所示。在神经网络中加入噪声后，输出的 Q 函数将随机化，相较于仅在智能体的动作上添加噪声，使得智能体的探索能力大大提高。

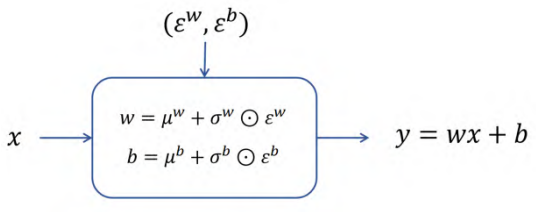


图 6　参数化噪声示意图
Fig.6　Parametric noise diagram

我们采用分解高斯噪声（Factorized Gaussian Noise）来生成噪声ε，假设全连接层有 m 个神经元，上一层有 n 个神经元，在这两层中，每个神经元都生成一个独立的高斯噪声，将两个不同神经元的噪声的乘积作为相应连接权重的噪音，如式(15)所示：

$$\begin{aligned} \varepsilon_{i,j}^{w} &= f(\varepsilon_i) f(\varepsilon_j) \\ \varepsilon_j^{b} &= f(\varepsilon_j) \end{aligned} \tag{15}$$

其中，$f(x) = sgn(x)\sqrt{x}$，分解高斯噪声最终会生成 m+n 个噪声，相对简洁。

在网络中引入噪声，虽然在一定程度上增加了参数的数量和计算开销，但一方面可以带来更大的探索效率，提高智能体的策略性能，另一方面，由于噪声的权重和标准差都是学习的参数，因此网络可以通过学习来调整噪声的大小，具有更强的适应性。

3.4　算法描述

SDQN 的算法伪代码描述如算法 1 所示，算法的工作原理如下：首先，初始化环境状态，将环境状态传入当前值网络（行 1-行 2）；其次，对于每个机器人，将除当前机器人外的其他机器人的历史地图和当前机器人的一系列视觉图像输入神经网络（行 3-行 5）；第三，在神经网络中引入 GRU 模块，并在全连接层的连接权重上添加参数化噪声，得到预测意图地图（行 6-行 8）；第四，将预测意图地图、当前机器人的历史地图和视觉图像一起输入神经网络，采用 Dueling 机制，输出随机化 Q 值，机器人选择相应的动作，得到新的状态和奖励值（行 9-行 12）；第五，将元组$(s_i^t, a_i^t, r_i^t, s_i^{t+1})$存入经验池，所有机器人共享同一个经验池，之后从经验池中随机取样（行 13-行 14）；第六，确定是否是一个 episode 的终止状态；如果是，时间差分目标（Temporal Difference Target，TD Target）为r_i^j；否则，用 TD 目标网络进行计算（行 15）；最后，使用梯度下降算法更新网络参数，每隔 N 步更新目标网络权重（行 16-行 18）。

算法 1: SDQN 运动规划算法

输入：经验池 D，经验池最大容量 N_r，初始化网络参数 θ，初始化目标网络参数 θ^-，batch_size N_b，目标网络更新频率 N^-，折扣因子γ

1. For episode = 1, M do
2. 　初始化环境状态：x ←env.reset ()
3. 　For t = 1, T do
4. 　　For each robot i do
5. 　　将 robot $j(1 \leqslant j \leqslant N,\ j \neq i)$的历史地图和 robot i 的环境顶视

图、智能体状态地图、最短路径地图输入神经网络

6. 在神经网络中加入 GRU 模块
7. 在神经网络全连接层的连接权重上添加参数化噪声
8. 输出预测意图地图
9. 将预测意图地图和 robot i 的历史地图、环境顶视图、智能体状态地图、最短路径地图输入神经网络
10. 将 Q 值分解为状态值和优势函数：式(14)
11. 输出随机化 Q 函数，机器人选择相应的动作a_i^t
12. 得到机器人执行动作a_i^t后的奖励r_i^t和下一个状态 s_i^{t+1}
13. 将当前转换元组$(s_i^t, a_i^t, r_i^t, s_i^{t+1})$存入经验池 $\mathcal{D}$，如果超过最大容量N_r，则替换最早的元组
14. 从经验池中随机采样N_b个元组，假设其中一个元组为$(s_i^j, a_i^j, r_i^j, s_i^{j+1})$
15. TD 目标为：

$$y_i^j = \begin{cases} r_i^j, & j+1 \text{ 是终止状态} \\ r_i^j + \gamma Q(s_i^{j+1}, \mathrm{argmax}_a Q(s_i^{j+1}, a; \theta, \alpha, \beta); & \\ \theta^-, \alpha, \beta), & j+1 \text{ 不是终止状态} \end{cases}$$

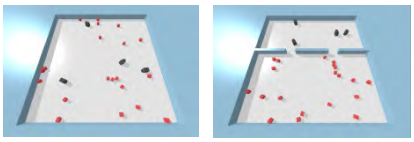
16. 对损失函数$(y_i^j - Q(s_i^j, a_i^j; \theta, \alpha, \beta))^2$执行梯度下降，得到$\nabla\theta$
17. 更新网络参数：$\theta = \theta + \nabla\theta$
18. 每隔N^-步，更新一次θ^-取值，令$\theta^- = \theta$
19. End for
20. End for
21. End for

4 实验与分析

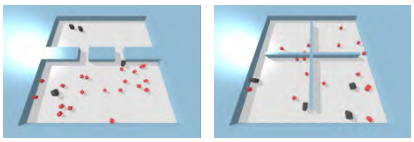
4.1 实验设置

实验环境为 Ubuntu 18.04 系统和 PyBullet 仿真平台。为了验证本文提出的 SDQN 方法的效果，我们根据场景大小及障碍物的复杂程度设置了 6 种不同的场景（基于文献[24]）。仿真场景如图 7 所示，包括 2 种小型场景和 4 种大型场景，每个场景均包含 4 个黑色机器人和若干个红色方块代表的目标点，其中，小场景包含 10 个目标点，大场景包含 20 个目标点。评价指标为固定时间内多机器人团队到达目标点的数量，通过标准差来表示策略的稳定性。

(a) SmallEmpty (b) SmallDivider



(c) LargeEmpty (d) LargeDoors



(e) LargeTunnels (f) LargeRooms

图 7 仿真场景图

Fig.7 Simulation scenes

4.2 实验细节

实验过程中，我们采用了文献[17]中的 MA$_2$IMP 方法，将历史地图和视觉数据相结合，共同预测智能体的意图。与 MA$_2$IMP 方法不同的是，本文采用 SDQN 方法进行策略训练，而不是简单地使用 DDQN 方法。

由于 SDQN 方法中，引入了基于噪声的有效探索机制，因此在实验过程中不再使用ε-greedy 策略进行探索，实验的具体参数如表 1 所示。

表 1 实验参数设置

Table 1 Experimental parameter setting

实验参数	值
batch size	32
折扣因子	0.35
学习率	0.01
权重衰减	0.0001
策略网络训练频率	4 个时间步
目标网络更新频率	1000 个时间步
训练开始时间比率	0.025

4.3 实验结果

4.3.1 主要结果

目前，关注多智能体之间意图的运动规划方法并不多，Jimmy Wu 等人[25]的空间意图方法最具代表性，相较于目前已有的多智能体运动规划方法，该方法的性能已经相当优越，因此，我们以文献[25]的空间意图方法作为基线来验证本文方法的有效性。

我们将本文所提的 SDQN 方法与基线 Spatial Intention 方法和 MA$_2$IMP 方法（我们的前期工作）进行对比，具体实验结果如表 2 所示，测试实验的结果曲线如图 8 所示。可以看出，在 6 种不同复杂程度的场景中，SDQN 方法的目标点到达数量均高于 MA$_2$IMP 方法和基线方法，标准差

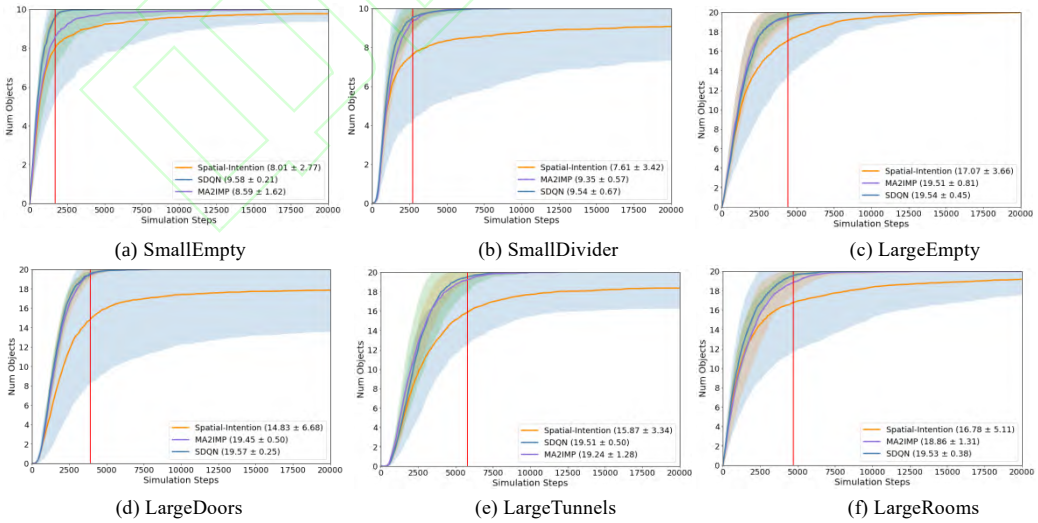
均低于 MA_2IMP 方法和基线方法。这说明本文所提出的 SDQN 方法可以有效地提升智能体运动规划策略的性能和稳定性。对 SDQN 方法的结果采取均值计算，可以得出，相较于基线 Spatial Intention 方法，SDQN 方法的策略性能平均提升了 17.17%。

在 6 种场景中，LargeRooms 场景中的性能提升非常显著，采用 MA_2IMP 模型时，多智能体团队成功到达目标点的数量均值为 18.86，标准差为 1.31，虽然相较于 Spatial Intention 方法已经有了明显提升，但是相较于 LargeEmpty、LargeDoors 和 LargeTunnels 几个场景，性能提升的幅度不太大，这说明 MA_2IMP 模型可能更适用于空旷环境和包含门、通道等瓶颈结构的场景，在包含房间结构的场景中的性能还有待提升。SDQN 方法使得 LargeRooms 场景中的目标点到达数量均值提升到 19.53，标准差降低到 0.38，已经达到甚至超越了在其他场景中的表现。我们对这一现象进行分析，推测其原因可能是，LargeRooms 场景由 4 个房间构成，智能体需要找到房间出口来到达隔壁房间中的目标点，完成这一任务存在较大的难度，智能体很可能由于找不到狭窄的出口而在同一个房间内来回移动。SDQN 方法中引入了基于 Dueling 的 Q 值计算优化机制，其中的优势函数可以衡量每个动作相对于其他动作有多好，从而找到更优的智能体动作；引入了基于 GRU 的记忆机制，对历史信息有记忆能力，从而有利于避免进行重复的移动；引入了基于噪声的探索机制，很好地提高了智能体的探索效率，使得智能体可以更多地探索之前未探索过的区域，从而更快寻找到房间出口。因此，相较于 MA_2IMP 模型，SDQN 方法更加适用于包含房间结构的场景，同时，在空旷环境和包含门、通道等瓶颈结构的场景中性能也较为优越。

表 2　SDQN 方法对比实验结果

Table 2　Comparative experimental results of SDQN

场景	SDQN	MA_2IMP[18]	Spatial Intention[25]
SmallEmpty	9.58±0.21	8.59±1.62	8.01±2.77
SmallDivider	9.54±0.67	9.35±0.57	7.61±3.42
LargeEmpty	19.54±0.45	19.51±0.81	17.07±3.66
LargeDoors	19.57±0.25	19.45±0.50	14.83±6.68
LargeTunnels	19.51±0.50	19.24±1.28	15.87±3.34
LargeRooms	19.53±0.38	18.86±1.31	16.78±5.11



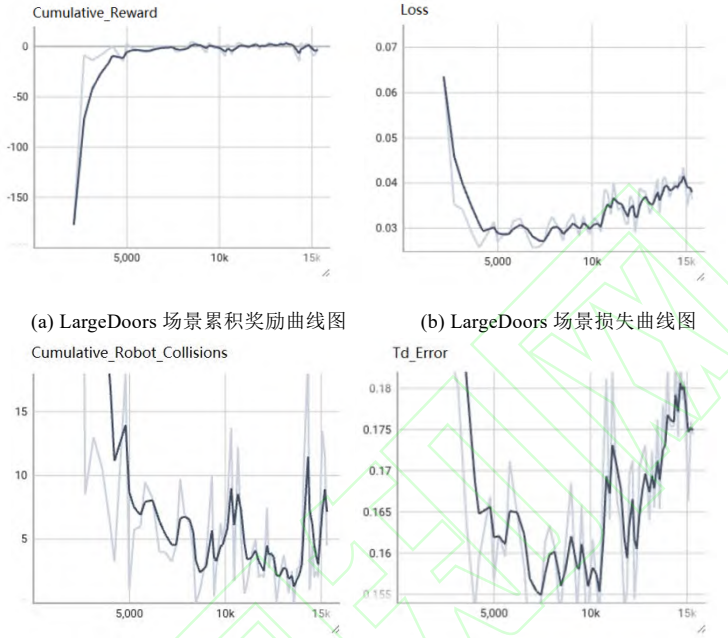
(a) SmallEmpty　(b) SmallDivider　(c) LargeEmpty

(d) LargeDoors　(e) LargeTunnels　(f) LargeRooms

图 8　6 种场景下目标点到达数量曲线图

Fig.8　Number of target points reached in six scenes

为了更加清晰地显示训练过程，我们以 LargeDoors 场景中的一次训练为例，绘制了训练过程中的累积奖励曲线（如图 9(a)所示）、损失函数曲线（如图 9(b)所示）、累积智能体碰撞曲线（如图 9(c)所示）以及 TD 误差曲线（如图 9(d)所示）。从图 9(a)和图 9(b)中可以看出，在训练到 5000 个时间步时，曲线开始趋于稳定，说明此时的策略雏形已经基本形成，后续还需要进行进一步的微调。图 9(d)反映了当更新策略网络的参数时，当前状态与预期状态的差异，可以看出，TD Error 在 0.175 左右达到稳定。



(a) LargeDoors 场景累积奖励曲线图　(b) LargeDoors 场景损失曲线图

(c) LargeDoors 场景智能体碰撞曲线图　(d) LargeDoors 场景 TD 误差曲线图

图 9　训练过程曲线图

Fig.9　Training process graph

4.3.2 消融实验

为了验证本文所提出的 SDQN 方法中的 3 个机制的有效性，我们针对本文所提的三种主要机制进行了消融实验。消融实验的详细结果如表 3 所示，其中，No-GRU 方法表示不采用基于 GRU 的记忆机制，在 DDQN 方法的基础上增加了基于噪声的有效探索机制和基于 Dueling 的 Q 值计算优化机制；No-Noisy 方法表示不采用基于噪声的有效探索机制，在 DDQN 方法的基础上增加了基于 GRU 的记忆机制和基于 Dueling 的 Q 值计算优化机制；No-Dueling 方法表示不采用基于 Dueling 的 Q 值计算优化机制，在 DDQN 方法的基础上增加了基于 GRU 的记忆机制和基于噪声的有效探索机制。

表 3　SDQN 方法消融实验结果

Table 3　Experimental results of ablation by SDQN method

场景	SDQN	No-GRU	No-Noisy	No-Dueling
SmallEmpty	9.58±0.21	8.55±0.68	8.63±1.06	8.91±1.29
SmallDivider	9.54±0.67	5.47±3.33	8.46±1.56	7.45±2.15
LargeEmpty	19.54±0.45	16.22±3.77	14.38±6.08	18.70±2.04
LargeDoors	19.57±0.25	11.24±5.96	16.86±1.38	14.81±3.25
LargeTunnels	19.45±0.56	15.55±7.80	17.30±2.36	17.97±3.53
LargeRooms	19.53±0.38	14.46±6.29	17.27±2.22	19.11±1.09

对消融实验结果进行直观展示，图 10 和图 11 的柱状图分别展示了目标点到达数量值和标准差值在不同场景、不同方法下的对比效果。可以看出，在 6 种场景中，SDQN 方法的目标点到达数量值均高于 No-GRU、No-Noisy、No-Dueling 3 种方法，标准差值均低于 No-GRU、No-Noisy、No-Dueling 3 种方法，这说明基于 GRU 的记忆机制、基于噪声的有效探索机制和基于 Dueling 的 Q 值计算优化机制对于多智能体团队的策略学习来说，都是十分必要的，缺少任何一种机制，都会对多智能体的合作产生不利影响。

从图 10 中可以看出，No-GRU 方法的效果表现最差，说明 GRU 模块的引入对于 SDQN 方法处理智能体的历史信息起到了关键作用；No-Dueling 方法的效果在三种对比方法中表现尚可，说明相对于基于 GRU 的记忆机制和基于噪声的有效探索机制来说，基于 Dueling 的 Q 值计算优化机制带来的系统性能增益较小。从图 11 中可以看出，No-GRU 方法产生的策略标准差值很大，说明基于 GRU 的记忆机制对于策略训练稳定性也起到了重要作用；No-Noisy 方法和 No-Dueling 方法产生的标准差值大致相当，说明基于噪声的有效探索机制和基于 Dueling 的 Q 值计算优化机制对于策略稳定性的增益效果大致相当。

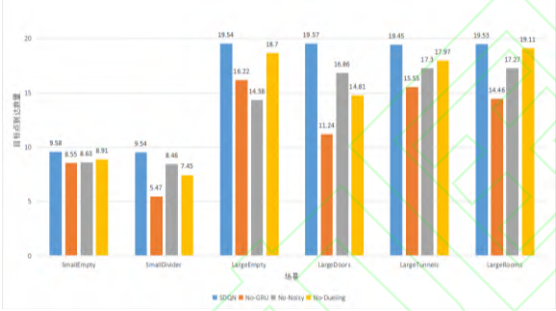


图 10　目标点到达数量值对比图

Fig.10　Comparison of the number of target points reached

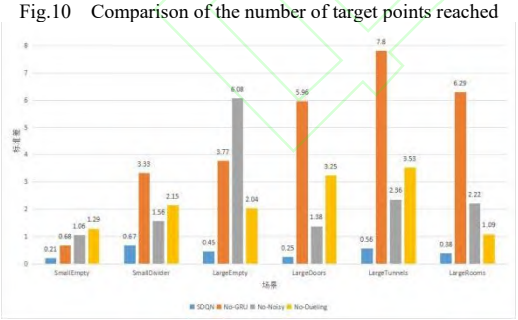


图 11　标准差值对比图

Fig.11　Standard deviation comparison chart

5　结　语

本文提出了一种基于 DQN 的多智能体深度强化学习运动规划方法 SDQN。SDQN 方法基于 DQN，引入了三个重要机制，其中，基于 Dueling 的 Q 值计算优化机制使得 Q 值的计算更加简单准确，基于 GRU 的记忆机制增强了 DRL 方法处理智能体历史信息的能力，基于噪声的有效探索机制提高了智能体的探索效率。仿真实验结果证明，该方法相较于传统的 DQN 方法，使多智能体团队到达目标点的数量平均提高了 17.17%，有效提升了多智能体团队的协作性能。在未来的工作中，我们将继续深入研究基于深度强化学习的协同运动规划问题，并将该运动规划算法移植到真实机器人上进行实物实验和验证。

参考文献

[1]　Hildebrandt A C, Klischat M, Wahrmann D, et al. RealTime Path Planning in Unknown Environments for Bipedal Robots[J]. IEEE Robotics and Automation Letters, 2017, 2(4): 1856－1863.

[2]　Holte R C, Perez M B, Zimmer R M, et al. Hierarchical A*: Searching abstraction hierarchies efficiently[C]//AAAI/IAAI, Vol. 1. 1996: 530-535.

[3]　Dorigo M, Maniezzo V, Colorni A. The ant system: An autocatalytic optimizing process[J]. 1991.

[4]　Khatib O. Real-time obstacle avoidance system for manipulators and mobile robots[C]//Proceedings of the 1985 IEEE International Conference on Robotics and Automation, St. Louis, MO, USA. 1985: 25-28.

[5]　MNIH V, KAVUKCUOGLU K, SILVER D, et al. Playing atari with deep reinforcement learning[J]. arXiv preprint arXiv:1312.5602, 2013.

[6]　Pu Zhiqiang, Yi Jianqiang, Liu Zhen, et al. A review of collaborative knowledge and Data driven swarm intelligent decision making [J]. Acta Automatica Sinica, 2022, 48(3): 1-17. (in Chinese)

蒲志强，易建强，刘振，等. 知识和数据协同驱动的群体智能决策方法研究综述[J]. 自动化学报, 2022, 48(3): 1-17.

[7]　Sharon G, Stern R, Felner A, et al. Conflict-based search for optimal multi-agent pathfinding[J]. Artificial Intelligence, 2015, 219: 40-66.

[8]　Fox D , Burgard W , Thrun S . The Dynamic Window Approach to Collision Avoidance[J]. IEEE Robotics & Automation Magazine, 2002, 4(1):23-33.

[9]　Gupta J K, Egorov M, Kochenderfer M. Cooperative multi-agent control using deep reinforcement

learning[C]//International conference on autonomous agents and multiagent systems. Springer, Cham, 2017: 66-83.

[10] Busoniu L, Babuska R, De Schutter B. Multi-agent reinforcement learning: A survey[C]//2006 9th International Conference on Control, Automation, Robotics and Vision. IEEE, 2006: 1-6.

[11] Hernandez-leal P, Kartal B, Taylor M E. A survey and critique of multiagent deep reinforcement learning[J]. Autonomous Agents and Multi-Agent Systems, 2019, 33(6): 750-797.

[12] Wang W, Yang T, Liu Y, et al. From few to more: Large-scale dynamic multiagent curriculum learning[C]//Proceedings of the AAAI Conference on Artificial Intelligence. 2020, 34(05): 7293-7300.

[13] Wang D, Deng H, Pan Z. Mrcdrl: Multi-robot coordination with deep reinforcement learning[J]. Neurocomputing, 2020, 406: 68-76.

[14] Wang D, Deng H. Multirobot coordination with deep reinforcement learning in complex environments[J]. Expert Systems with Applications, 2021, 180: 115128.

[15] Sunehag P， Lever G， Gruslys A，et al. Value-Decomposition Networks For Cooperative Multi-Agent Learning[J]. 2017.

[16] Mnih V, Kavukcuoglu K, Silver D, et al. Playing atari with deep reinforcement learning[J]. arXiv preprint arXiv:1312.5602, 2013.

[17] Van Hasselt H, Guez A, Silver D. Deep reinforcement learning with double q-learning[C]//Proceedings of the AAAI conference on artificial intelligence. 2016, 30(1).

[18] Peng Yingxuan, Shi Dianxi, Yang Huanhuan, HU Haomeng, Yang Shaowu. Motion planning Method for Multi-agent Deep Reinforcement Learning Based on Intention [J]. Computer Science, 2023-10.

彭滢璇, 史殿习, 杨焕焕, 胡浩萌, 杨绍武. 基于意图的多智能体深度强化学习运动规划方法[J]. 计算机科学, 2023-10.

[19] Wang Z, Schaul T, Hessel M, et al. Dueling network architectures for deep reinforcement learning[C]//International conference on machine learning. PMLR, 2016: 1995-2003.

[20] Hausknecht M, Stone P. Deep recurrent q-learning for partially observable mdps[C]//2015 aaai fall symposium series. 2015.

[21] Yao S, Chen G, Pan L, et al. Multi-robot collision avoidance with map-based deep reinforcement learning[C]//2020 IEEE 32nd International Conference on Tools with Artificial Intelligence (ICTAI). IEEE, 2020: 532-539.

[22] Sukhbaatar S, Fergus R. Learning multiagent communication with backpropagation[J]. Advances in neural information processing systems, 2016, 29.

[23] Liu Y, Wang W, Hu Y, et al. Multi-agent game abstraction via graph attention neural network[C]//Proceedings of the AAAI Conference on Artificial Intelligence. 2020, 34(05): 7211-7218.

[24] Mahajan A, Rashid T, Samvelyan M, et al. Maven: Multi-agent variational exploration[J]. Advances in Neural Information Processing Systems, 2019, 32.

[25] Wu J, Sun X, Zeng A, et al. Spatial intention maps for multi-agent mobile manipulation[C]//2021 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2021: 8749-8756.

史殿习：出生于 1966 年，博士，研究员，博士生导师，IEEE 会员。主要研究方向为人工智能、机器人操作系统、分布式计算及云计算等。

彭滢璇：出生于 1998 年，硕士研究生。主要研究方向包括人工智能、多智能体协同、强化学习、机器学习等。

杨焕焕：出生于 1994 年，博士研究生。主要研究方向包括人工智能、强化学习、机器学习等。

欧阳倩滢：出生于 1998 年，硕士研究生，主要研究方向为目标跟踪和强化学习等。

张玉晖：出生于 1987 年，硕士，工程师。主要研究方向为人工智能、路径规划和多智能体协同。

郝　锋：出生于 1977 年，硕士，副研究员，主要研究方向为机械电子工程、计算机应用。

SHI Dian-Xi, born in 1966, PhD, professor, PhD supervisor, is a member of IEEE. His main research interests include artificial intelligence, robot oper-ating system , distributed computing and cloud computing.

HAO Feng, born in 1977, Master, associate professor. His main research interests include artificial intelligence, mechanical and electronic Engineering and computer Applications.

文章自校联系人：彭滢璇　　邮箱：pengyingxuan98@163.com