分类号_____ 密级_____

UDC_____ 编号_____

全日制专业硕士学位论文

基于强化学习的车辆边缘计算任务调度算法研究

学位申请人： 晏云辉

专业领域： 通信工程（含宽带网络、移动通信等）

校内导师： 聂学方 副教授

答辩日期：2023年6月3日

基于强化学习的车辆边缘计算任务调度算法研究

摘要

随着 5G 通信速率的大幅提升和城市化的飞速发展，大量时延敏感型车联网应用应运而生，这些时延敏感型应用对计算资源的需求较高，然而车载终端计算资源有限，无法实现任务实时处理。针对以上问题，车辆边缘计算（Vehicle Edge Computing,VEC）是一个有效的解决方案。VEC 将服务器部署在靠近车辆的路侧，为车辆提供所需的计算资源。然而网络运营商部署的 VEC 服务器计算资源有限，在上下班等交通高峰期，车载终端卸载的任务将大幅增加，服务器计算负载将临近饱和状态，导致任务处理时延明显增加，这会使交通存在安全隐患。针对该问题，实施基于 VEC 服务器的高效任务调度策略，提高任务处理效率是可行的解决方案。基于此，本文以降低任务处理时延和提高任务处理成功概率为目标，研究如何用深度强化学习（Deep Reinforcement Learning, DRL）来解决 VEC 服务器计算任务调度问题。论文的具体工作如下：

（1）为解决车辆边缘计算网络在交通高峰时段的任务处理效率下降问题，本文提出基于 VEC 服务器的任务处理框架，构建多智能体强化学习任务调度模型，并用多智能体深度 Q 网络（Multi Agent Deep Q-Networks, MADQN）算法求解任务调度策略，以达到提高任务实时处理效率、降低 VEC 任务处理时延的目的。在 MADQN 任务调度策略中，嵌入基于线程服务时间的奖励机制，保障深度神经网络快速收敛。同时引入多智能体集中训练和分布式运行的机制，降低计算维度，减少深度神经网络的训练时间，以适应车联网实时任务处理环境。实验结果表明，相比于其它任务调度算法，本文提出的 MADQN 任务调度算法可获得更低的任务处理时延和更高的任务处理成功概率。

（2）为进一步提高任务处理效率，利用多智能体双重深度 Q 网络（Multi Agent Double Deep Q-Networks, MADDQN）算法对提出的 MADQN 算法进行优化。其中 MADQN 算法在计算 Q 值时存在过高估计问题，MADDQN 算法针对该问题，使用双重 Q 网络计算 Q 值，有效的减少算法的估计偏差，从而进一步提高算法的性能。实验结果表明，相比于 MADQN 任务调度算法，MADDQN 任务调度算法可获得进一步的性能提升。

关键词：任务调度，MADQN，MADDQN，车辆边缘计算，车联网

Research on Task Scheduling Algorithms for Vehicle Edge Computing Based on Reinforcement Learning

Abstract

With the significant increase in 5G communication rate and the rapid development of urbanization, a large number of latency-sensitive telematics business applications have emerged. These latency-sensitive applications have a high demand for computing resources. However, the limited computing resources of vehicle terminals cannot realize the real-time processing of tasks. For the above problems, Vehicle Edge Computing (VEC) is an effective solution. VEC deploys servers on the roadside close to the vehicle to provide the required computing resources for the vehicle. However, network operators have limited computing resources to deploy VEC servers, and during peak commuting traffic periods, the tasks offloaded from vehicle terminals will increase significantly, and the server computing load will be close to saturation, leading to a significant increase in task processing latency and traffic safety hazards. To address this problem, implementing an efficient task scheduling strategy based on VEC server to improve task processing efficiency is a feasible solution. Based on this, this article aims to reduce task processing latency and improve task processing success probability, and investigates how to utilize Deep Reinforcement Learning (DRL) to solve the VEC server computing task scheduling problem. The specific work of the paper is as follows.

(1) To solve the problem of decreasing task processing efficiency of vehicle edge computing networks during peak traffic hours, this paper proposes a task processing framework based on VEC, constructs a multi-agent reinforcement learning task scheduling model, and solves the task scheduling strategy using Multi-Agent Deep Q-Networks (MADQN) algorithm. The MADQN algorithm is used to solve the task scheduling policy in order to improve the real-time processing efficiency and reduce the processing delay of VEC tasks. In the MADQN task scheduling strategy, a reward mechanism based on thread service time is embedded to guarantee the fast convergence of deep neural networks. The mechanism of multi-agent centralized training and distributed operation is also introduced to reduce the computational dimensionality and the training time of deep neural networks to adapt to the real-time task processing environment of VEC. The experimental results show that the MADQN task scheduling algorithm proposed in this paper can obtain lower task processing latency and higher task processing success probability compared with other task scheduling algorithms.

(2) To further improve the task processing efficiency, the proposed MADQN algorithm is optimized using the Multi Agent Double Deep Q-Networks (MADDQN) algorithm. Among them, the MADQN algorithm has an overestimation problem in calculating Q values. The MADDQN algorithm addresses this problem by using a double Q-network to calculate Q values,

which effectively reduces the estimation bias of the algorithm and thus further improves the performance of the algorithm. Experimental results show that the MADDQN task scheduling algorithm can obtain further performance improvement compared to the MADQN task scheduling algorithm.

Key words: task scheduling, MADQN, MADDQN, vehicle edge computing, internet of vehicle

目　录

第一章　绪论

1.1　研究背景和意义

随着城市化的蓬勃发展及 5G 通信速率的大幅度增加，智能交通系统（Intelligent Transportation System, ITS）获得了汽车工业界和学术界的广泛关注。由于 ITS 技术利用现代通信技术和信息技术对道路进行智能控制，并且采用了科学的交通流量控制，能够大大提高道路交通的稳定性，从而改善城市交通的质量。而近年来，装有人机交互技术装置以及大量感应器的智慧车辆也已日益增多。这些车辆配备了多种功能模块，有传感器模块（如高清摄像头）、无线电设备模块（如 IEEE 802.11p）、车载单元模块（如 On Board Unit, OBU）以及其他功能模块，这些功能模块一部分负责感知周围环境，一部分负责通信和计算[1]。通信基础设施也正在向智能化方向发展,例如路侧单元（Road Side Unit, RSU）和微基站，这些设备将沿着道路安装并和骨干网络相连接。这些通信基础设施通过和车辆进行数据传输，可以为车辆提供各种服务。通过车辆和基础设施之间的相互连接，形成了大规模的网络，这就是车联网。作为 ITS 的关键部分，车联网通过车辆与基础设施间通信（Vehicle to Infrastructure, V2I）和车辆与车辆间通信（Vehicle to Vehicle, V2V）等方式，为车载终端提供多种多样的需求信息服务，如紧急信息广播（例如重大交通事故）和资讯获取服务（例如娱乐新闻）。车联网多样的通信信息服务使车辆出行更加的安全和舒适，据前瞻产业研究院预测，到 2025 年，全球将拥有约 3.8 亿辆联网汽车，如图 1-1 所示[2]。

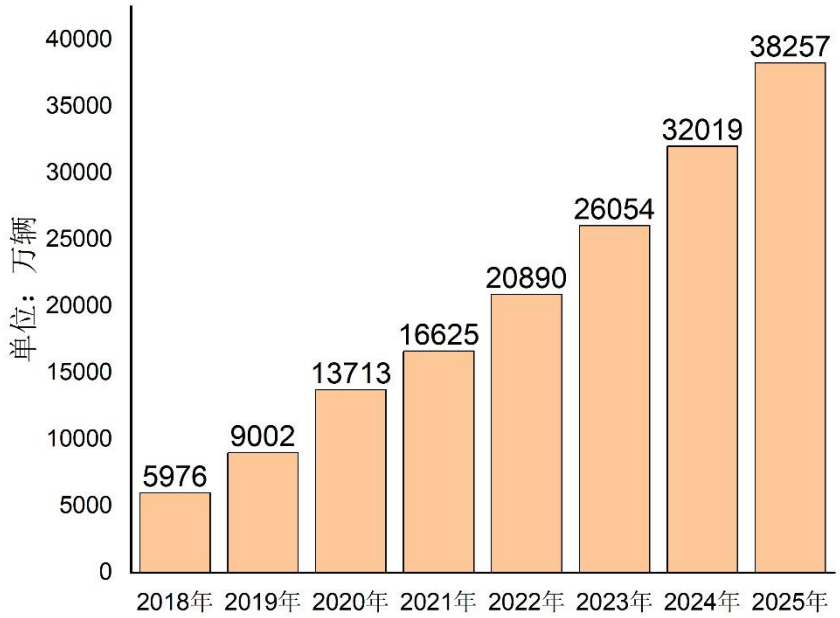

图 1-1　车辆统计与预测

Fig.1-1　Vehicle statistics and forecasting

随着车联网的快速发展，智能车辆也得到了更广泛的应用，包括自动驾驶和智能泊车技术等[3]。这些新运用都是计算资源密集型和时延敏感型应用，为满足这类新兴应用需要大量的计算资源和高效的任务处理能力。而车载终端则需要实时处理来自不同传感器的大量数据并做出正确的决策[4]。然而，车载终端装备的功能模块所能提供的计算资源和计算能力有限，不足以支持计算资源密集型和时延敏感型应用[5]。这主要是由于汽车经济成本和功能模块安装空间的限制所导致的。随着车联网应用的不断演进，传统的车联网架构面临着越来越大的挑战，即如何满足不断增长的存储和计算资源需求。

为解决车载终端数据存储和任务计算资源不足的问题，研究者采用了云计算技术来对车辆进行集中性的存储和计算服务。云计算技术在汽车互联中的主要意义就在于，一方面车辆能够直接把计算密集型应用卸载至云端加以管理，从而免除了庞大的运算和储存需求；另一方面，云服务器还能够存放来自各个车辆互联实体的信息，从而使得车辆能够从任意位置存取信息，进而达到了车辆之间的信息资源共享[6]。在车联网领域，基于云的卸载架构已经成为车辆执行运算和信息储存的主要解决方案。云计算技术能够显著降低车辆的信息储存和运算压力，被看作是未来车联网的一项主要趋势。然而，车联网云服务器一般配置在车辆远离的位置。当处理信息通过车载终端传输至云服务器并将处理好的信息回复至车载终端时，需要通过各种设备实现转发，涉及 RSU、基站以及互联网，并且远距离传输导致传输延迟较高，使云服务器无法满足延迟敏感类服务的需求[7]。此外，随着用户数量和汽车数量的增加，车联网中的信息数据流出现了指数式的增长。而一旦将所有的信息数据流全部都传输至核心网云服务器进行处理，将导致核心网回程链路过载，导致车联网网络严重拥堵，最终降低了整个车联网网络性能和车辆的服务质量。基于以上分析，表明云计算无法完全满足计算资源密集型和时延敏感型应用的需求[8,9]。

车联网发展需求促进了网络数据处理能力和智能化向数据源更接近的方向发展，在这种趋势下，车辆边缘计算（Vehicular Edge Computing, VEC）应运而生。VEC 通过将边缘计算技术引入到车联网中，解决了云计算核心网回程链路过载问题。在 VEC 中，通过将远程云服务器的部分存储和计算资源移动到更靠近车辆的网络边缘[10]，车载终端只需进行单跳无线访问，即可使用车联网网络的计算资源，而无需通过多跳再访问云服务器，大大减少了数据传输时间。VEC 的优势主要有两方面，一方面可以减少车联网核心网回程链路上的数据流量传输，以避免车联网网络拥堵；另一方面可以显著减少车联网应用的响应时间，提高用户满意度[11]。基于以上优势，VEC 作为车联网领域的重点发展方向，在提高车辆安全、提高交通舒适度及提升交通质量等方面有着极大的发展潜力[12]。

随着计算资源密集型和时延敏感型应用的不断增加，为提高车辆用户的服务质量，VEC 变得更为重要。但是，网络运营商基于成本考虑，部署 VEC 服务器的计算资源有限[13]。在上下班等交通高峰期，车载终端卸载的任务将大幅增加，服务器计算负载将临

近饱和状态，导致任务处理时延明显增加，存在较大的交通安全隐患。基于此，本文将致力于研究车联网边缘计算中的任务密集型调度场景，提出了基于深度强化学习的任务调度算法，以期在提升任务处理效率和减少任务处理时延方面发挥重要作用。

1.2 国内外研究现状

1.2.1 车联网研究现状

近年来，车联网已成为世界各国政府和企业关注和投资的焦点，被视为一项战略性产业。通过将人工智能技术、5G 技术和传统汽车产业相融合，车联网技术正在加速发展，引领着汽车产业的新变革。在车联网技术发展的过程中，各国积累了大量的经验和技术，不断创新和改进车联网技术的各个方面，以提高车辆的智能化、互联化和安全性。同时，各国通过收集和分析车联网产生的大量数据，为车联网产业提供了有力支持。

在国际层面，美国从 2015 年开始发布车联网智能道路交通系统战略规划，以促进车联网智能道路交通行业的发展。在欧洲，通过一系列法律和监管文件，包括《道路交通法》第 8 条修订案等，帮助车联网相关企业自由测试和应用新技术。此外，欧盟 2019 年还发布了新规定，为车联网智能交通系统确定政策，以促进欧盟整体投资和监管框架，以整合智能车辆和车联网交通基础设施。日本政府将发展无人驾驶汽车技术放在首位，对无人驾驶进行了多项测试，并引入了一系列制度和法律制度，以促进智能汽车产业的发展。

在国内层面，车联网智能汽车产业已成为我国竞争日益激烈的重要产业。我国明确了汽车向智能化和网联化的方向协同发展，并制定了具体的车联网智能汽车发展路线图。从 2017 年开始，我国陆续发布了一系列车联网智能汽车产业政策和法规，以推动产业快速发展。2020 年，政府多个部门联合发布了车联网产业管理文件，为未来车联网产业的发展规划提供了指导。此外，国内各城市和企业高度重视车联网智能汽车产业的发展，并积极推进相关技术的应用和研发。互联网产业也开始加入车联网领域。华为成立了智能汽车解决方案 BU，向智能驾驶领域进军。百度推出了 CarLife＋平台，并在多个城市展开无人驾驶测试，同时推出 Apollo 开发平台。另一个互联网巨头腾讯也推出了车联网系统“All in Car”和车辆自动驾驶 TAD Sim2.0 虚拟仿真平台。此外，众多科技企业也投入了大量精力和资源，发展新一代 C-V2X 技术，为迎接未来车联网的发展机遇做好准备[14]。

1.2.2 车辆边缘计算研究现状及面临的挑战

1.2.2.1 车辆边缘计算研究现状

目前，车联网中的车辆边缘计算已经受到汽车产业和学术界的广泛重视，并产生了大量的研究成果。下面，本节将综述车联网中车辆边缘计算的研究现状。

已经有多项研究成果关于车联网中的车辆边缘计算卸载架构。其中，文献[15]采用云计算中心来为车辆提供计算任务卸载服务。当大量车辆的请求计算任务卸载时，云服务器由于距离的原因无法满足需求，为此文献[16]提出了一种基于云计算和车辆边缘计算协作的任务卸载架构。文献[17]则将车辆与车联网云服务器相结合，提出了一种基于云计算和车载终端计算资源相互协作的车联网计算任务卸载架构。文献[18]提出了一种三层车联网计算任务卸载架构，该架构由云计算服务器层、边缘计算服务器层和用户层组成，可以将计算任务卸载至边缘服务器中进行计算。

在车联网车辆边缘计算任务卸载的性能指标方面，也有多项研究成果。例如，文献[19]通过对车辆用户侧和边缘服务器侧的缓存资源和计算资源进行合理分配来实现最小化车联网系统能耗的目标。文献[20]则考虑了服务器处理计算任务和车辆与基站间通信的能耗，利用人工鱼群算法来使车联网的计算卸载能耗最小化。文献[15]提出了联合车辆云和中心云的车联网计算任务卸载方案，以最小化车联网系统的时延、资源和能耗等。对于时延敏感型的任务，文献[21]提出了基于 VEC 的软件定义网络体系架构，以提升车联网的服务质量。

针对车联网车辆边缘计算任务卸载问题，许多传统的优化算法，如博弈论和启发式优化算法等，已被广泛应用。针对车辆竞争式地将需要进行计算的任务卸载到车联网服务器的情况，文献[22]使用博弈论对其进行建模，同时加入分布式激励机制，以降低车辆之间的竞争性。在基于云计算服务卸载的场景中，文献[23]使用图论求解最短路径问题。文献[24]使用改进的遗传算法来管理车联网计算任务卸载，以实现系统总时延最小化的目标。文献[25]使用多臂赌博机来解决车联网动态变化的问题。然而，上述研究只考虑了简单的车联网应用场景，不适应于更加复杂的车辆边缘计算环境。因此，需要进行更深入的研究，以满足车联网的发展需求。

1.2.2.2　车辆边缘计算面临的挑战

目前，基于车联网的车辆边缘计算面临着诸多挑战。

（1）在车联网中，存在着资源不确定性的问题，这会影响到车辆边缘计算任务卸载对资源环境感知的能力。在车辆边缘计算中，车辆是否进行计算任务卸载存在主观性差异。由于隐私敏感等原因，许多车辆不愿意分享自己的计算资源，导致车辆间资源状态不能共享。而未知的资源状态会影响到车辆边缘计算任务卸载的资源环境感知能力，从而导致计算任务卸载的可靠性降低。

（2）车联网中存在着资源动态变化问题，这会导致车辆边缘计算任务卸载环境的不稳定性。作为车联网场景的重要特点，动态变化性表现在车辆位置、拓扑结构和通信环境等方面。目前，大多数车辆边缘计算任务卸载的研究都假设车联网计算资源相对稳定。而在现实下，车辆驾驶状态、路况和天气等因素会导致计算资源状态发生明显的变

化。计算资源动态变化性会导致车联网计算任务卸载的环境发生变化，严重影响车辆边缘计算卸载环境适应性。

（3）在车联网中，存在着多任务高并发性问题，这会导致车辆边缘计算任务卸载环境变得更加复杂。随着人工智能技术的不断发展，大量新兴车联网应用应运而生。在为车辆用户提供丰富的车联网服务的同时，必须确保行车安全。未来，多任务高并发性的车联网场景将随处可见，这将导致车联网中计算资源的激烈竞争，同时加剧了资源的管理与优化。因此，车辆边缘计算任务卸载环境变得更加复杂将成为一项巨大的挑战，影响计算任务卸载的运算效率[14]。

根据上述分析，车联网中的计算任务卸载面临的问题对传统优化算法提出了挑战。由于传统优化算法的固定模型和高计算复杂度等特点，使传统优化算法难以解决车辆边缘计算面临的资源不确定、资源动态变化和多任务高并发问题。为了解决这些问题，本文将深度强化学习算法（Deep Reinforcement Learning, DRL）引入到车辆边缘计算任务调度问题中。DRL 算法可以感知环境的变化，并对各种环境状态做出决策，能满足各种车联网场景的需求。下面将介绍 DRL 算法在车辆边缘计算研究现状。

1.2.3　基于深度强化学习的车辆边缘计算研究现状

随着 DRL 算法理论的不断完善，DRL 已广泛应用于车联网中车辆智能控制、交通流量控制、计算资源管理与优化等方面[26]。DRL 算法可以感知环境的变化，并对各种环境状态做出决策，能满足各种车联网场景的需求。DRL 被认为是解决车辆边缘计算的有效手段[27]。下面，将介绍 DRL 在车辆边缘计算的研究现状。

目前已有众多学者运用 DRL 方法解决车联网问题。如文献[28]提出了一种分层式的结构，以应对高维环境状态空间及高维智能体动作空间问题，解决了云计算中车联网资源分配及发射功率优化问题。文献[29]为实现资源管理的智能化，提出了一种基于 DRL 的车联网资源管理方案。文献[30]为实现具有依赖性计算任务卸载，提出了一种基于 DRL 的车联网资源管理和任务调度方案。文献[31]则提出了一种基于 DRL 的车联网公平性资源分配方案，以解决车联网中资源分配的问题。文献[32]为解决 VEC 任务调度场景任务处理时延和系统能耗最小化问题，利用马尔可夫决策过程（Markov decision process, MDP）建立任务调度模型，通过 DRL 方法优化任务调度方案，实验验证了 DRL 方法用于解决复杂决策问题的有效性。文献[33]利用车联网中 RSU 和车载计算资源，实现车辆计算任务卸载，其中 RSU 为车辆提供通信功能和计算资源，车辆协作进行数据传输和计算。文献[34]针对车路云协同场景，运用 DRL 方法优化计算资源联合调度方案，降低任务处理时延。文献[35]提出了一种基于 DRL 算法的方案，考虑了车辆和路边单元 RSU 作为移动边缘计算节点和基站，实现最小化能量消耗和数据传输时延的目标。文献[36]研究了车联网中车辆和边缘计算的协作式任务卸载策略，车辆可以将还未计算完成的计算卸

载任务迁移到附件的其他空闲车辆，以完成任务计算。文献[37]利用 DRL 算法为车联网求解联合车载计算资源及边缘服务器计算资源的计算卸载策略，以最小化系统能耗。

但是，车联网 VEC 规模不断扩大，任务计算量呈指数增长，传统 DRL 算法无法满足高维度计算需求[38]。基于 DRL 的多智能体处理采用集中训练分布式运行方式，大幅降低计算资源开销，适用于执行车联网复杂环境下的决策任务。文献[39]采用 DRL 方法优化车辆到车辆（Vehicle to Vehicle, V2V）无线链路的频谱分配方案，并执行多智能体分布式任务处理方式，实现网络信道总容量的提升。文献[40]提出多智能体强化学习及拉格朗日乘子两级嵌套的组合优化方法，解决车载终端任务卸载及服务器缓存优化问题，从而提高系统效能。文献[41]设计多智能体 DRL 算法解决车联网车载终端任务卸载最优决策问题，最小化任务执行时延。

然而，这些研究主要关注车载终端的任务卸载决策问题，并假设 VEC 服务器计算资源充足，车载终端卸载的计算任务均可得到实时处理。事实上，VEC 服务器受成本和硬件资源的限制，可提供的计算资源相对有限。在高峰期 VEC 服务器接收任务过于密集而处于饱和状态，从而延长任务处理时延、降低系统可靠性。因此，充分利用有限的计算资源，研发高效的任务调度算法提高任务执行效率亟待解决。

1.3　论文的主要内容和结构安排

本文首先介绍了车联网的快速发展，车联网通过将人工智能技术、5G 技术和传统汽车产业相融合，催生了大量新型计算密集型和时延敏感型运用。为了满足这些新型应用，VEC 通过将服务器部署到更靠近车辆的网络边缘，可以更好的为车辆提供计算资源，大大减少了数据传输时间。然而网络运营商部署的 VEC 服务器计算资源有限，为保证交通畅通和交通安全，急需一个高效的车辆边缘计算任务调度策略。且由于车联网场景中计算任务问题的复杂性和车联网络的动态性，使传统优化算法在实现车联网中计算任务卸载方面存在较大的限制。为此，本文将 DRL 算法引入车联网中，用 DRL 算法解决车辆边缘计算任务调度问题。针对 VEC 任务密集型的任务调度场景，提出基于 VEC 服务器的任务处理框架，并将多任务调度多线程的场景建模为多智能体强化学习问题，再用多智能体深度强化学习算法求解任务最优调度策略，以提升任务处理效率和减少任务处理时延。

本文的创新点一是针对车联网 VEC 任务密集型的任务调度场景，本文将调度场景建模成多智能体强化学习问题，并提出多智能体深度 Q 网络（Multi Agent Deep Q-Networks, MADQN）算法求解任务最优调度策略，以提升任务处理效率和减少任务处理时延。为提升 MADQN 算法的效率，在算法训练阶段，提出了基于线程时间的奖励机制，使任务调度优化问题与任务处理时延优化问题相结合；在算法执行阶段，提出分布式运行方式，以减少任务处理时延。

本文的创新点二是在车联网 VEC 任务密集型的任务调度场景下，为了解决深度 Q 网络（Deep Q-Networks, DQN）算法对 Q 值偏高估计的问题，本文提出了基于多智能双重深度 Q 网络（Multi Agent Double Deep Q-Networks, MADDQN）任务调度算法，以优化算法对 Q 值的估计，从而进一步减少 VEC 服务器任务处理时延，提高 VEC 车辆的服务质量。

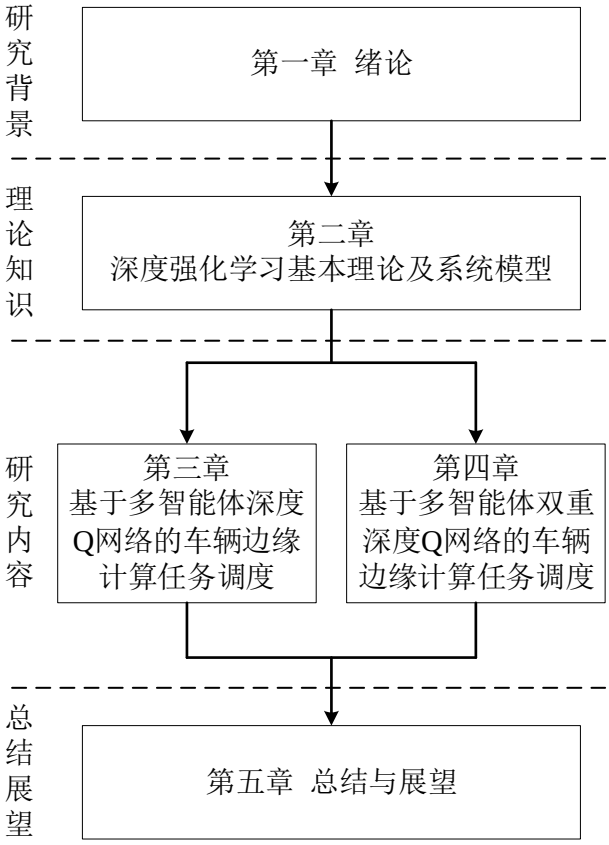

图 1-2　论文结构安排

Fig.1-2　Paper structure arrangement

本文的内容总共分为五章，论文结构安排如图 1-2 所示。

第一章绪论。主要是介绍车联网技术的发展背景，并对车联网的研究现状，车辆边缘计算研究现状及面临的挑战和基于深度强化学习的车辆边缘计算研究现状进行概括。以及介绍本文的主要内容和结构安排。

第二章介绍深度强化学习基本理论及本文的系统模型。理论部分本文介绍了马尔可夫决策过程、强化学习算法、深度强化学习算法和多智能体强化学习。最后介绍本文 VEC 密集型任务调度场景系统模型，并提出了优化问题。

第三章本文的主要研究内容之一，基于多智能体深度 Q 网络的车辆边缘计算任务调度。针对车联网 VEC 密集型任务调度场景，提出 MADQN 算法求解任务最优调度策略。在算法训练阶段，提出了基于线程时间的奖励机制，使任务调度优化问题与任务处

理时延优化问题相结合；在算法执行阶段，提出分布式运行方式，以减少任务处理时延。最后给出实验仿真验证了所提算法的有效性。

第四章本文的主要研究内容之二，基于多智能体双重深度 Q 网络的车辆边缘计算任务调度。针对在车联网 VEC 密集型任务调度场景下，DQN 算法对 Q 值偏高估计的问题，本文提出了基于 MADDQN 的任务调度算法。MADDQN 通过使用双重 Q 网络结构，将 Q 估计交由评估网络和目标网络共同完成，有效地减少过高估计问题，从而提高算法的收敛速度和性能。最后给出实验仿真验证了 MADDQN 算法优于 MADQN 算法。

第五章总结与展望。总结全文的研究内容和主要贡献，以及后续需要深入研究的方向。

第二章　深度强化学习基本理论及系统模型

动物或者人天生就具备智慧的大脑，我们将其叫做天然智能（Natural Intelligence），而智能机器通过各种设备配合展现出的智慧，我们将其叫做人工智能（Artificial Intelligence, AI）。机器学习（Machine Learning, ML），它可以让机器配备各种智能设备，并不断训练，能使机器具备一定的学习能力，所以人们认为 ML 是一种重要的实现人工智能的方式[42]。强化学习（Reinforcement Learning, RL），是机器学习中的一种主要的学习模式。RL 的作用在于可以为智能体做出决策，智能体在与环境的交互过程中，通过不断试错的方式学习，而 RL 就是为智能体判断哪些方式是对的，哪些方式是错的。RL 判断的依据就是这个方式能否使回报最大化或达到预定的目标[43]。深度学习（Deep Learning, DL）是机器学习的另外一种重要的学习模式，DL 通过使用多层神经网络来模拟人的大脑，不断训练神经网络，使机器具有一定的学习能力。近年来随着 CPU、GPU 等硬件计算能力的发展，DL 在图像识别、语音交互和自动驾驶等新兴应用领域取得了引人注目的表现[44]。因此，人们渴望 RL 能够和 DL 融合，并用来解决之前棘手的难题。2015 年，谷歌团队在 Nature 期刊上发表了一篇用深度强化学习（Deep Reinforcement Learning, DRL）算法来训练机器玩 Atari 游戏的文章，实验结果显示，机器通过 DRL 的学习，可以达到和人游玩时一样的水平，这篇文章向人们展示了 DRL 算法的巨大发展潜力[45]。

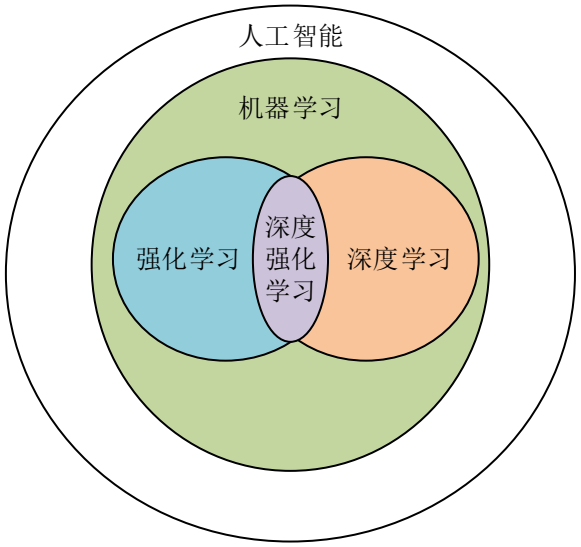

图 2-1　机器学习的学习模式

Fig.2-1　Learning Mode of Machine Learning

图 2-1 展示了，强化学习、深度学习和深度强化学习的关系，三者都是机器学习的子集。本章节首先将简要介绍深度强化学习理论和多智能体强化学习理论，最后介绍深度强化学习任务调度模型和优化问题。

2.1　深度强化学习理论

2.1.1　马尔可夫决策过程理论

在智能体和环境不断交互的过程中，智能体可以通过观测环境，获得当前的环境状态，之后智能体根据当前观测到的环境状态进行决策并做出一个动作。智能体执行动作后，环境会因此而发生改变到下一个环境状态，并将执行动作的奖励反馈给智能体。智能体和环境不断重复这个过程，直到环境达到终止状态或达到指定的回合数才停止。这个过程中，智能体需要学习如何使奖励最大化。下图 2-2 展示了智能体-环境交互过程。我们使用马尔可夫决策过程（Markov Decision Process, MDP）对其进行建模：一个 MDP 用一个五元组表示：$(\mathcal{S},\mathcal{A},\mathcal{R},\mathcal{P},\gamma)$，其中 $\mathcal{S}$ 代表的是环境状态空间，环境状态空间 $\mathcal{S}$ 包含环境的各个因素；$\mathcal{A}$ 代表的是智能体动作空间，智能体动作空间 $\mathcal{A}$ 包含了智能体对环境执行的各种动作；$\mathcal{R}$ 代表的是奖励函数，奖励函数 $\mathcal{R}$ 是智能体执行所选动作后，环境改变所获得的奖励；$\mathcal{P}$ 代表的是状态转移函数，状态转移函数 $\mathcal{P}$ 是当前环境状态改变到下一个环境状态时，到下一个环境状态的概率；γ 为奖励折扣因子，奖励折扣因子 γ 是未来奖励对当前奖励的影响。将智能体-环境交互时间离散化为 $0,1,\cdots,t,\cdots$，智能体-环境交互的过程可以描述为：在某一时刻 t，智能体观测环境，获得环境状态 $S_t\in\mathcal{S}$，智能体根据状态 S_t 执行动作 $A_t\in\mathcal{A}$，智能体执行动作 A_t 后，环境转移到下一个状态 $S_{t+1}\in\mathcal{S}$，并将奖励 R_{t+1} 反馈给智能体。之后智能体与环境之间不断交互，直到环境达到终止状态或达到指定的回合数才停止。

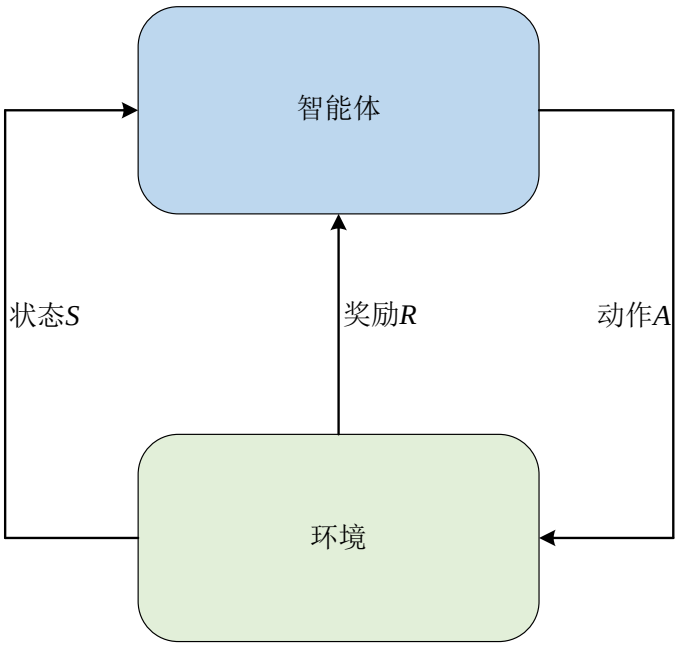

图 2-2　智能体-环境交互过程

Fig.2-2　Agent-Environment Interaction Process

2.1.1.1 状态和状态空间

在上面介绍的 MDP 中，环境状态空间$\mathcal{S}$代表的是一个有限的非空集合，假设环境状态空间有N个智能体动作，则环境状态空间可表示为$\mathcal{S}=\{s_1,s_2,\cdots,s_n\}$。环境状态空间中的元素是环境信息的总和，每一个元素都是环境信息的抽象化。如车联网中基站和车辆形成的环境状态，包括基站的频谱和计算资源的状态，汽车的位置和速度等状态信息。在时刻t的环境状态可以用随机变量$S_t \in \mathcal{S}$来表示，具体的值可以用s_t表示。

2.1.1.2　动作和动作空间

智能体动作空间$\mathcal{A}$可以用一个有限非空集合表示，假设智能体动作空间包含K个动作，那么$\mathcal{A}$可以表示为$\mathcal{A}=\{a_1,a_2,\cdots,a_k\}$。智能体动作空间中的元素是智能体对环境状态执行可以影响时其发生变化的行为总和，$\mathcal{A}$中的元素都是抽象的。如车联网频谱共享中，车辆选择频谱的行为就是车辆对车联网频谱共享环境执行的动作，或者车联网任务卸载中，车辆选择将任务卸载到边缘基站中进行计算，车辆选择卸载任务的行为就是车辆对车联网任务卸载环境做出的动作。在时刻t智能体动作可以用随机变量A_t表示，具体的值可以用a_t表示。

2.1.1.3　状态转移概率

智能体在环境状态S_t下，根据S_t做出决策，再执行动作A_t，环境在智能体施加的动作影响下发生改变，进入到下一个状态S_{t+1}。其中，环境从S_t变化到S_{t+1}是根据概率变化的，我们用状态转换函数$\mathcal{P}$来描述环境状态转移过程中，状态转移的概率$p(s_{t+1}|s_t,a_t)$。显而易见，状态转移函数$\mathcal{P}$应该满足约束：$0\le\mathcal{P}(s_t,a_t,s_{t+1})\le 1$。如果所有环境状态和智能体动作都是离散的，那么就有$\sum_{s_{t+1}\in\mathcal{S}}\mathcal{P}(s_t,a_t,s_{t+1})=1$，这说明环境从$S_t$变化到$S_{t+1}$的概率累加为 1。

如果环境状态的转移只和当前环境状态和智能体动作有关，而与在这之前的所有环境状态和智能体动作都无关，那么我们就认为这个交互过程具有马尔可夫性[46]，即

$$\Pr(S_{t+1}|S_t,A_t,S_{t-1},A_{t-1},\cdots)=\Pr(S_{t+1}|S_t,A_t) \tag{2-1}$$

马尔可夫性的关键就是，智能体根据当前状态的信息可以做出最优决策，而不需要在这之前的环境状态和智能体动作信息。如车辆边缘计算中，智能体接收到环境状态信息S_t，执行动作A_t后，环境以概率$p(s_{t+1}|s_t,a_t)$变化到下一个状态S_{t+1}。其中概率$p(s_{t+1}|s_t,a_t)$就是车辆边缘计算的状态转移概率。状态从s_t变化到状态s_{t+1}有多个不同的状态变化，所以一般用概率$p(s_{t+1}|s_t,a_t)$来表示其状态变化的可能性。

2.1.1.4　奖励和奖励函数

奖励函数表示，智能体根据观测到的当前环境状态 S_t 做出决策，并执行动作 A_t 后，环境状态变化到下一个环境状态 S_{t+1}，环境发生状态变化并将奖励反馈给智能体，这个奖励由奖励函数 R_t 产生。我们将奖励函数 $R_t(s_t,a_t)$ 定义为

$$R_t(s_t,a_t)=\mathbb{E}\left[R_t\middle|S_t=s_t,A_t=a_t\right]=\sum_{R\in\mathcal{R}}R\sum_{s_{t+1}\in\mathcal{S}}p(s_{t+1},R|s_t,a_t) \tag{2-2}$$

分析公式（2-2）可知，在环境状态 s_t 下，智能体执行动作 a_t，可以使环境状态的变化有多种可能 $s_{t+1}\in\mathcal{S}$，所以我们求的奖励是所有可能状态的期望。在 t 时刻，智能体可以获得奖励 R_t，R_t 是一个随机变量，具体的值可以用 r_t 表示。

奖励函数是强化学习中的核心部分，因为奖励函数为智能体确定了学习的目标。智能体在学习的过程中需要学习到时使奖励最大化的策略。如车联网频谱分配中，车辆需要学习到选择频谱的策略，并使车联网系统的信道容量最大化。奖励函数的确定决定了马尔可夫决策过程的前进方向。

2.1.1.5　智能体的策略

智能体的策略可以用 $\pi(\cdot)$ 表示，它代表了一个从环境状态到智能体动作的映射。智能体观测到环境状态 s，并利用策略函数计算出需要执行的动作：$a=\pi(s)$。在马尔可夫决策过程中，如果智能体选择动作的策略是随机的，那么我们可以将其称为随机策略，即 $\pi(s,a)=\Pr(A_t=a|S_t=s)$。在随机策略中，每一个环境状态 $s\in\mathcal{S}$ 都有一个对应的动作集 $\mathcal{A}$。智能体的学习就是策略的学习，策略会根据积累的经验数据不断学习，一个训练好的策略，可以直接体现智能体执行能力的好坏。

2.1.1.6　智能体的目标

在强化学习中，智能体的目标是通过不断学习，从而找到一种最优策略，这个策略可以使累积奖励和期望最大化。对于一个智能体和环境交互的轨迹 $S_0,A_0,R_1,S_1,A_1,R_2,\cdots$，我们将智能体在 t 时刻之后可以获得的累积奖励称为回报，用 G_t 表示，可以定义为

$$G_t=R_{t+1}+\gamma R_{t+2}+\gamma^2R_{t+3}+\cdots=\sum_{k=0}^{\infty}\gamma^kR_{t+k+1} \tag{2-3}$$

其中 $\gamma(0\le\gamma\le1)$ 为折扣因子。由公式（2-3）可知，未来奖励离当前时间的距离越远，产生的折扣越多。也就是说未来奖励离当前时间的距离越远，对当前时刻奖励的影响越小。具体当折扣因子 $\gamma=0$，智能体会只在意当前时刻的奖励，而不考虑未来奖励，这样智能体会变得自顾眼前利益；而当折扣因子 $\gamma=1$，智能体会完全在意未来的奖励，这会时算法的收敛性无法得到保证。因此，强化学习的折扣因子在设置大小时，一般将其值设计为接近 1。

2.1.2 强化学习算法

强化学习的目标是使智能体通过不断学习，从而找到一个最优策略π^*。智能体在最优策略π^*的决策指导下，可以获得所有环境状态最大化的预期回报，π^*可以表示为

$$\pi^* = \arg\max_{\pi} \mathbb{E}\left[G_t \middle| \pi\right] \tag{2-4}$$

根据公式（2-3）回报函数G_t的定义，可以定义强化学习的价值函数。强化学习用于让智能体通过与环境的交互来学习如何做出最优决策。为了在给定状态下做出最优决策，智能体需要对其行动的后果进行评估。这就需要估计一个价值函数，该函数可以评估智能体在给定状态或状态-动作对下的表现好坏。智能体的表现好坏程度是通过预期回报来衡量的，即智能体在未来可能获得的奖励。这些奖励是由行动策略来决定的，行动策略是智能体在每个状态下采取的行动方式。因此，为了获得最优回报，智能体必须学习如何选择最优的行动策略。因此，强化学习价值函数的定义依赖于特定的策略。接下来本文将给出具体定义。

状态价值函数：强化学习的状态价值函数$V_\pi(s)$表示从环境状态s开始，智能体按照策略π选择动作，执行所选动作后所获得的预期回报，状态价值函数$V_\pi(s)$可以表示为

$$V_\pi(s) = \mathbb{E}_\pi\left[G_t \middle| S_t = s\right] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \middle| S_t = s\right] \tag{2-5}$$

由公式（2-5）可知，智能体可以用状态价值函数$V_\pi(s)$来判断当前环境状态s的价值。

状态-动作价值函数：强化学习的状态-动作价值函数$Q_\pi(s,a)$表示在环境状态s下，智能体执行动作a后，智能体再按照策略π选择后续的动作，这种情况下智能体所获得的预期回报就是状态-动作价值$Q_\pi(s,a)$，状态-动作价值函数$Q_\pi(s,a)$可以表示为

$$Q_\pi(s,a) = \mathbb{E}_\pi\left[G_t \middle| S_t = s, A_t = a\right] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \middle| S_t = s, A_t = a\right] \tag{2-6}$$

由公式（2-6）可知，智能体可以用状态-动作价值$Q_\pi(s,a)$来判断当前环境状态s下执行动作a的价值。

强化学习的状态-动作价值函数和状态价值函数，可以用 Bellman 期望方程来展示两个价值函数之间相互转换关系，下面将具体展示：

用强化学习的状态-动作价值函数来表示其状态价值函数，如下所示

$$V_\pi(s) = \sum_a \pi(a|s) Q_\pi(s,a), s \in \mathcal{S} \tag{2-7}$$

由公式（2-7）可知，强化学习的状态价值可以由当前状态s下，所有状态-动作价值的期望来表示。

用强化学习的状态价值函数来表示其状态-动作价值函数，如下所示

$$\begin{aligned} Q_\pi(s,a) &= R(s,a)+\gamma\sum_{s'} p(s'|s,a)V_\pi(s') \\ &= \sum_{s',R} p(s',R|s,a)\left[R+\gamma V_\pi(s')\right], s\in\mathcal{S}, a\in\mathcal{A} \end{aligned} \tag{2-8}$$

由公式（2-8）可知，强化学习的状态-动作价值可以由当前状态s下并执行动作a后，所获得的即时奖励和后续所有状态价值的期望来表示。

基于公式（2-7）和式（2-8），我们使用代入法可以消除公式中的一种强化学习价值函数。消除之后即可获得值函数的递归表示，具体表示如下：

用强化学习状态价值函数来表示其状态价值函数：

$$V_\pi(s)=\sum_a \pi(a|s)\left[R(s,a)+\gamma\sum_{s'} p(s'|s,a)V_\pi(s')\right], s\in\mathcal{S} \tag{2-9}$$

由公式（2-9）可知，强化学习的状态价值可以由当前状态s下，执行不同动作a后，所获得的立即奖励和后续所有状态价值的期望来表示。

用强化学习状态-动作价值函数表示状态-动作价值函数：

$$Q_\pi(s,a)=\sum_{s',R} p(s',R|s,a)\left[R+\gamma\sum_{a'}\pi(a'|s')Q_\pi(s',a')\right], s\in\mathcal{S}, a\in\mathcal{A} \tag{2-10}$$

由公式（2-10）可知，强化学习的状态-动作价值可以由当前状态s下并执行动作a后，变化到各种状态s'，将各种状态所获得的立即奖励和后续所有状态-动作价值累加，并将其求期望的值来表示。

在强化学习环境动力$p(s',R|s,a)$已知的情况下，就可以根据公式（2-9）和公式（2-10），利用数学动态规划的方法，用一个函数来无限逼近在给定了智能体策略π情况下，强化学习的期望值函数。由于强化学习值函数反映了在给定策略π情况下的预期期望值，因此可根据强化学习值函数的大小来判断最优的策略。首先定义强化学习的最优状态价值函数$V^*(s)$

$$V^*(s)=\max_\pi V_\pi(s), s\in\mathcal{S} \tag{2-11}$$

即在状态s下，执行最优策略，使智能体获得最大的状态价值。

强化学习最优状态-动作价值函数$Q^*(s,a)$

$$Q^*(s,a)=\max_\pi Q_\pi(s,a), s\in\mathcal{S}, a\in\mathcal{A} \tag{2-12}$$

即在当前状态s下执行动作a，之后再执行最优策略，使智能体获得最大的状态-动作价值。

对于一个特定 MDP，最优策略可能有多个且不同的策略。根据上述的强化学习价值函数的定义，可知即使最优策略不同，但是这些最优策略所取得的价值函数的值是相同的，因此可以随机选择一个最优策略进行考察。其中一种强化学习最优策略为，根据下面展示的确定性策略进行智能体动作的选择

$$\pi^*(s)=\arg\max_{a\in\mathcal{A}} Q^*(s,a), s\in\mathcal{S} \tag{2-13}$$

即在当前状态s下，按照策略π^*执行动作a，使智能体获得的状态价值函数最大化。

Q-learning 算法[47]是 RL 最基础的算法。Q-learning 算法是一种基于列表型的 RL 算法。Q-learning 算法使用价值函数$Q(s,a)$来代表智能体在环境状态s下执行动作a所获得奖励的期望值，也称为Q值。Q值包括环境立即奖励和环境未来的奖励，并通过将Q值存储到 Q-learning 列表中，以此来实现Q值存储和更新。然而，由于状态空间和动作空间的不断增大，存储空间有限的列表型 Q-learning 算法不再适用。而且，由于组合(s,a)的多样性，列表型 Q-learning 算法需要对每一组(s,a)组合进行多次访问，这样智能体才能进行有效学习。这样的学习使智能体学习过程效率低下，无法应对高维、连续动作空间的问题。因此，为了克服这些问题，下面将介绍深度强化学习算法，深度强化学习算法是由强化算法和深度学习算法结合组成的。

2.1.3 深度强化学习算法

传统的强化学习算法，如 Q-learning 算法，因为其存储列表的约束只能处理简单的任务环境。一旦处理现实中复杂的任务，这些复杂任务的环境状态空间和智能体动作空间通常都是连续的，这时再使用传统的强化学习算法将使智能体无法学习到最优策略。目前，随着深度学习理论的发展，人们开始研究如何将深度学习和强化学习结合起来，形成深度强化学习算法来解决上述传统强化学习算法无法解决的问题。接下来介绍深度强化学习中基于值函数近似的深度 Q 网络算法（Deep Q network, DQN）[45]及双重深度 Q 网络算法（Double Deep Q network, DDQN）[48]。

2.1.3.1 深度 Q 网络算法

在强化学习算法中，许多环境状态都具有很大且连续的环境状态空间，如果此时直接用值函数来估计离散的状态-动作价值函数会使估计值不准确。为此，可以使用一个参数化的连续函数$Q(s,a;\omega)$来逼近要估计的状态-动作价值函数[45]。连续函数$Q(s,a;\omega)$的参数ω是一个有限维向量，用随机数来初始化参数ω。为使连续函数$Q(s,a;\omega)$可以近似状态-动作价值函数，算法通过最小化损失函数$L(\omega)$来无限逼近状态-动作价值函数。损失函数$L(\omega)$可以表示为

$$L(\omega)=\mathbb{E}_{(s,a,R,s')}[(U^Q-Q(s,a;\omega))^2] \tag{2-14}$$

其中目标状态-动作价值函数$U^Q=R+\gamma\max_{a'}Q(s',a';\omega)$。

面对更加复杂的环境状态时，连续函数$Q(s,a;\omega)$需要评估网络（evaluation network）来对值函数进行近似。但是，在实际使用中，由于强化学习在训练过程中按顺序产生的大量训练数据存在较强的相关性，使评估网络在近似值函数时不够稳定或值离散。为了解决数据相关性过大的问题，DeepMind 提出了 DQN 算法[45]。DQN 将深度学习与 Q-learning 算法进行结合，用深度神经网络来估计值函数，可以实现更复杂的控制任务。

如在许多 Atari 游戏中，DQN 算法训练的机器人可以超过一般的人类玩家水平。此外，与 Q-learning 算法相比较，DQN 算法有两个关键的核心技术：

（1）第一个关键核心技术是在 DQN 算法中使用了固定 Q 值的目标网络（target network）。目标网络的结构与评估网络的结构一样。使用 DQN 算法时，公式（2-14）中的目标 Q 值 U^Q 的公式需修改为 U^{DQN}，U^{DQN} 可以表示为

$$U^{DQN} = R + \gamma \max_{a'} Q(s', a'; \omega') \tag{2-15}$$

其中 ω' 是目标网络的参数，在 DQN 算法训练过程中，在一定的训练迭代次数后，更新目标网络的参数 ω'，更新方式为从评估网络中复制评估网络参数 ω。在 DQN 算法中使用目标网络可以提高 DQN 算法的稳定性。图 2-3 展示了 DQN 中评估网络与目标网络之间的结构关系。

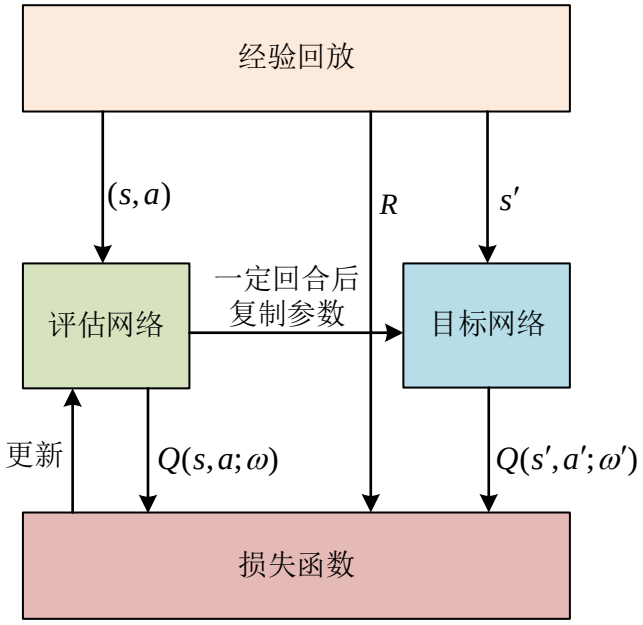

图 2-3　DQN 中评估网络与目标网络关系图

Fig.2-3　The relationship between the evaluation network and the target network in DQN

（2）第二个关键核心技术是在 DQN 算法中使用了经验回放结构（experience replay memory）。DQN 算法将训练过程中产生的经验数据 (s_t, a_t, R, s_{t+1}) 存储到经验回放 $\mathcal{K}$ 中，使智能体能够记忆之前的经验并重复使用过去的数据信息进行学习。在 DQN 算法训练过程中，经验回放池中任意两个数据元组之间是弱相关的，DQN 网络参数通过随机采样的方式从经验回放池中随机选取一批量经验数据来进行学习训练。通过从经验回放池中随机采样，打破了经验数据间的强相关性，同时能够确保系统能够稳定的收敛。因此，DQN 算法中损失函数 $L(\omega)$ 可以表示为：

$$L(\omega) = \mathbb{E}_{(s,a,R,s')\sim\mathcal{K}}[(U^{DQN} - Q(s,a;\omega))^2] \tag{2-16}$$

DQN 算法评估网络参数 ω 的迭代规则为：

$$\omega \leftarrow \omega + \mathbb{E}_{(s,a,R,s')\sim\mathcal{K}}[(R + \gamma \max_{a'} Q(s', a'; \omega') - Q(s, a; \omega))\nabla_{\omega} Q(s, a; \omega)] \tag{2-17}$$

2.1.3.2 双重深度 Q 网络算法

在 DQN 算法中，通过取最大化Q值的操作来选择动作，并用这个最大化的Q值计算状态-动作价值函数，这很容易使目标网络产生过高的Q值估计。为解决 DQN 算法过高估计问题，在文献[48]中，作者提出了双重深度 Q 网络算法 DDQN。在 DDQN 算法中，目标Q值和目标网络及评估网络都有关。目标Q值首先由评估网络选择使Q值最大的动作a，再由目标网络根据输入的s'和a计算目标Q值。DDQN 通过使用双网络结构，能够有效地减少过高估计问题。DDQN 的网络结构及网络参数更新过程与 DQN 算法类似，除了计算目标U^{DQN}的公式被改为U^{DDQN}。U^{DDQN}可以表示为

$$U^{DDQN} = R + \gamma Q(s', \arg\max_{a} Q(s', a; \omega); \omega') \tag{2-18}$$

2.2 多智能体强化学习理论

DRL 及 RL 学习，通常都用单个智能体来解决独立问题。但在现实世界中，很多现实问题并不是孤立的，这些问题往往需要多个子问题一起并行处理。这些问题就是多智能体强化学习的问题。例如，智能小车快递分配、车辆无人驾驶等，这些智能的现实应用场景中需要大量的智能体协作以实现总体目标。由于智能体的数量太多，往往不清楚特定智能体的行为是积极的还是消极的。因此，将环境中的其他智能体行为看作影响环境的因素，这时单个智能体的动作选择主要取决于其他智能体的行为，这就使智能体学习策略具有挑战性。因此，多智能体强化学习（Multi Agent Reinforcement Learning, MARL）是解决这种现实中多智能体应用场景的有效途径。

MARL 的应用非常广泛，可以用于无人驾驶汽车、交通控制、多人游戏、机器人控制、社会网络分析等领域。在这些场景中，多智能体必须根据环境和其他智能体的状态和行为做出独立的决策，不仅要最大化自己的奖励，也要为设定的总体目标做出贡献。通过使用 MARL 技术，可以实现更高效、更智能、更灵活的系统，从而为社会带来更多的好处和便利。因此，MARL 的研究可能会在未来几年内发挥重要作用。

MARL 的定义：多个智能体同时学习策略来解决自身问题，多个智能体最终实现解决一个复杂现实问题的过程，如图 2-4 所示。这种使用多个智能体来学习策略解决问题的方式就是 MARL[49]。在多智能体系统中，强化学习和协同决策是非常重要的。多个智能体需要根据环境和其他智能体的状态和行为做出独立的决策，以最大化自己的奖励或为总体目标做出贡献。这种情况下，每个智能体的决策都会影响整个系统的性能，因此，要取得良好的系统性能，需要对智能体之间的协同行为进行精心设计和调整。下面介绍 MARL 理论的组成部分。

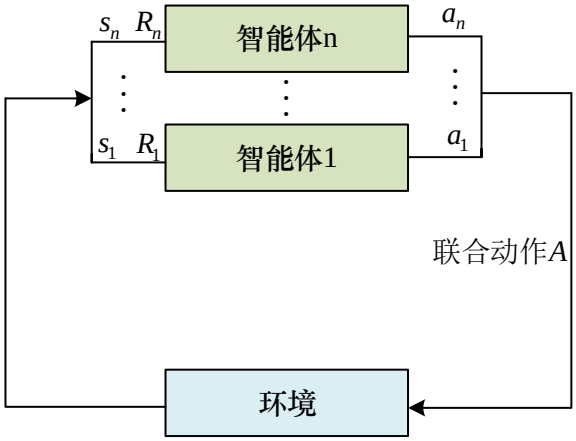

图 2-4　多智能体示意图

Fig.2-4　MARL schematic

2.2.1　随机博弈

随机博弈（Stochastic Game, SG）也称作马尔可夫博弈。SG 可以表示为$\left(S, A^n, R^n, P\right)$，其中 S 是状态空间； n 表示智能体的数量； A^n 是 n 个智能体的联合动作空间集合 $A^n = A_1 \times \cdots A_n$； R^n 表示智能体的奖励函数 $R^n = (R_1, R_2, \cdots, R_n)$； P 表示环境状态转移概率函数 $P \in [0,1]$。

按照环境任务类型的不同，基于随机博弈的 MARL 算法可以分为三种模式，分别是完全合作模式、完全竞争模式及混合合作竞争模式。

（1）完全合作模式：每个智能体的奖励函数都一样 $R_1 = R_2 = \cdots = R_n$，所有智能体的目标是一样的，为使共同回报最大化。我们称之为完全合作模式。

（2）完全竞争模式：如果智能体的数量 $n = 2$，然而，智能体的奖励函数 R_1 和 R_2 相反，奖励函数 $R_1 = -R_2$。我们称之为完全竞争模式。

（3）混合合作竞争模式：这种模式下，存在不是完全竞争模式的策略，也存在不完全合作模式的策略，所有智能体的奖励不受约束。

在完全合作模式下，MARL 算法智能体的目标是最大化共同回报，并解决单个智能体的优化问题。在完全竞争模式和混合合作竞争模式下，MARL 算法智能体的目标相对较难确定，多个智能体的奖励函数相互关联，使回报难以最大化。这时 MARL 算法的目标是收敛多个智能体的均衡点。本文使用的是多智能完全合作模式，下面将介绍本文的系统模型部分。

2.3　深度强化学习任务调度模型

本文考虑异构 VEC 密集型任务调度应用场景，如图 2-5 所示。宏基站连接的 VEC 服务器作为一个数据中心，负责模型训练。每个路侧单元（Roadside Unit, RSU）配置一

个 VEC 服务器，并通过光纤连接。假设每个 VEC 服务器有 M 个线程，并用集合 $\mathcal{M}=\{1,2,...,M\}$ 表示。线程 $m(1\le m\le M)$ 的计算资源用 f_m 表示，单位为 GHz/s。车载终端卸载的计算任务服从均匀分布[50]，用集合 $\mathcal{D}=\{1,2,\cdots,K\}$ 表示 VEC 服务器接收的任务集。每个任务 $k(1\le k\le K)$ 由任务数据大小 d_k（单位为 MB）和时延约束 τ_k（单位为 ms）构成。其中 d_k 和 τ_k 服从均匀分布。

当 VEC 服务器线程处于工作状态，任务处理时延需要加上线程等待时间，线程等待时间用集合 $\mathcal{T}=\{t_1,t_2,\cdots t_M\}$ 表示，$\mathcal{T}$ 服从高斯分布。用 $x_{k,m}$ 表示任务 k 的目标线程是线程 m，$x_{k,m}=1$ 就表示任务 k 选择线程 m 进行处理，否则 $x_{k,m}=0$。假设每个计算任务 k 只能选择一个线程，即 $\sum_{m\in\mathcal{M}}x_{k,m}=1$。本文假设每个任务 k 对目标线程的选择可以表示为智能体 k 对 VEC 服务器环境所执行的动作。VEC 任务调度场景如图 2-6 所示。

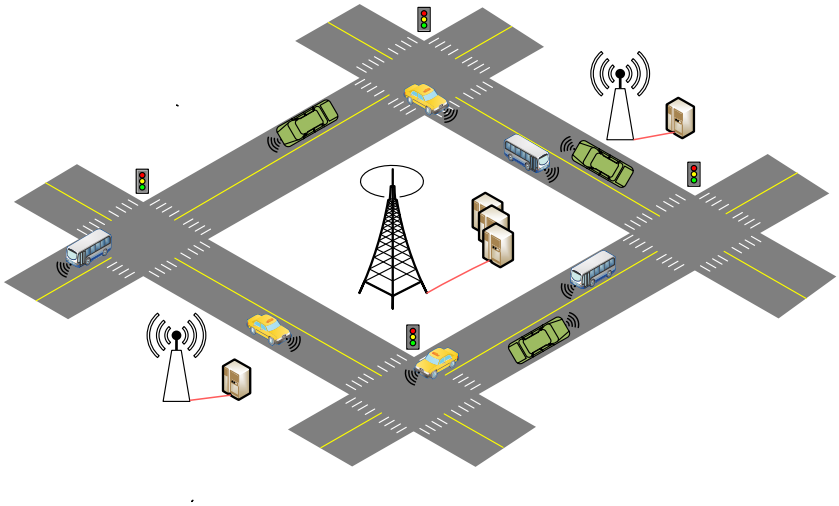

图 2-5　VEC 密集型任务调度场景

Fig.2-5　VEC-intensive task scheduling scenarios

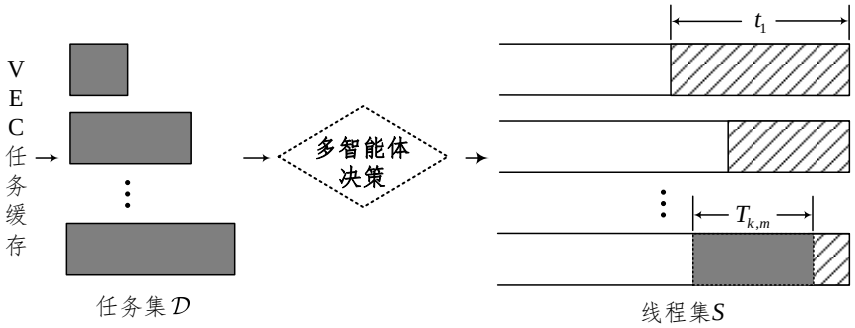

图 2-6　VEC 任务调度

图 2-6　VEC task scheduling

2.3.1 任务计算模型

如果任务k选择线程m进行处理，则任务k的处理时延$T_{k,m}$可表示为：

$$T_{k,m} = x_{k,m} \times d_k \times e / f_m \tag{2-19}$$

其中e表示线程的计算强度，单位为GHz/MB，e是一个常量，由硬件配置决定。由公式（2-19）可知，影响任务计算时延$T_{k,m}$的主要因素是任务大小d_k和线程计算资源f_m。

用T_m表示线程m的处理时延，T_m可表示为：

$$T_m = t_m + \sum_{k \in \mathcal{D}} T_{k,m} \tag{2-20}$$

其中t_m表示当前线程m等待时间。公式（2-20）表明线程m接收的任务越多，则线程任务处理时延T_m的值越大。

VEC服务器任务处理总时延T_s为线程集$\mathcal{M} = \{1,2,...,M\}$中任务处理时延最大项，即

$$T_s = \max\{T_1,...T_m,...,T_M\}. \tag{2-21}$$

由公式（2-21）可知T_m是影响VEC服务器计算总时延T_s的关键因素。

2.3.2 问题描述

VEC 服务器任务调度优化总目标为在保证各计算任务时间约束的基础上最小化VEC服务器任务处理总时延，因此优化问题可表述为

$$\begin{aligned} &\mathrm{P1}. \min_{\{k,m\}} \{\max_{m \in \mathcal{M}} T_m - \min_{m \in \mathcal{M}} T_m\} \\ &s.t. \\ &C_1 : T_{k,m} \le \tau_k \\ &C_2 : x_{k,m} \in \{0,1\} \\ &C_3 : \sum_{m \in \mathcal{M}} x_{k,m} = 1 \end{aligned} \tag{2-22}$$

其中C1表示任务计算时延约束，C2表示任务k对线程m的选择情况，C3表示一个任务只能分配到一个线程中进行处理。由于优化问题是一个混合整数的问题，所以优化问题是NP-hard。

2.4 本章小结

本章详细介绍了马尔可夫决策过程的基本理论、强化学习算法和深度强化学习算法，并介绍了DQN和DDQN两种深度强化学习，紧接着介绍了多智能体强化学习理论，最后介绍了本文构建的任务调度模型及优化问题。本章所介绍的内容将为后续算法的提出提供借鉴并奠定基础。

第三章　基于多智能体深度 Q 网络的车辆边缘计算任务调度

3.1　引言

本章对 VEC 密集型任务场景中的计算任务调度进行研究。随着 5G 通信速率的大幅提升，大量时延敏感型业务应运而生，如 AR/VR、实时导航和自动驾驶等[51,52]。在车联网自动驾驶场景中，目标检测、信息感知及智能决策等技术均需较高的计算资源支持[53]。然而车载终端计算资源有限，无法实现任务实时处理[54]。而计算资源充足的中央云处理，由于远距离数据传输将导致较大的通信时延，不适用于时延敏感型业务[9]。车辆边缘计算（Vehicle Edge Computing, VEC）将服务器部署于邻近车辆的路侧，通过光纤连接至路侧基站，避免了计算任务的远距离传输，可有效降低任务处理时延[55,56]。

基于成本考虑，网络运营商部署 VEC 服务器的计算资源有限[57]。在上下班交通高峰期，车载终端卸载的任务将大幅增加，服务器计算负载将临近饱和状态，导致任务处理时延明显增加，存在交通安全隐患。针对该问题，实施基于 VEC 服务器的高效任务调度策略提高任务处理效率是可行的解决方案[58]。为此，本文针对车联网 VEC 服务器密集型多任务调度场景，提出了基于多智能体深度 Q 网络（Multi Agent Deep Q-Networks, MADQN）的任务调度算法，以提升任务处理效率和减少任务处理时延。

3.2　基于多智能体的任务调度

第二章中介绍的 VEC 任务调度场景中，多个任务选择 VEC 服务器线程进行处理，可以建模成多智能体强化学习问题。每个任务充当一个任务智能体，并与 VEC 服务器环境交互获得经验，然后用于指导自己的调度策略学习。多个任务智能体共同探索 VEC 服务器环境，并根据自己对 VEC 服务器环境的观察，优化任务调度策略。虽然任务调度问题可能表现为一个竞争博弈，但是通过让所有智能体使用相同的奖励函数，可以将其转化为一个完全合作的博弈，提高网络的整体性能。

本文提出的基于多智能体深度 Q 网络方法分为两个阶段，即训练阶段和实施阶段。在训练阶段，每个任务智能体都可以获得 VEC 网络以计算效率为导向的奖励，然后通过更新其深度 Q 网络（Deep Q-Networks, DQN）将其行为调整为最优策略。在实施阶段，每个任务智能体观测局部环境状态，然后根据其训练的 DQN 选择一个动作。下面详细描述基于多智能体深度 Q 网络任务调度设计的关键要素。

3.2.1　状态和观测空间

在多智能体任务调度中，每个任务作为一个智能体，合作探索 VEC 服务器环境[24]。从数学上讲，该问题可以建模为 MDP。如图 3-1 所示，给定 t 时刻 VEC 环境状态 s_t，

每个任务智能体可以通过观测函数 O 确定 VEC 服务器环境状态 $Z_t^{(k)}=O(S_t,\ k)$，然后根据策略 π 选取动作 $A_t^{(k)}$，再形成联合动作 $\boldsymbol{A}_t$。之后，VEC 服务器环境以概率 $p(s^{'},R_t\mid s,a)$ 改变到下一个状态 S_{t+1}，并向每个任务智能体返回奖励 R_{t+1}。再之后任务智能体观测到新的状态 $Z_{t+1}^{(k)}$，再次与环境进行交互学习，不断试错，试图寻找最优策略，使其折扣累计奖励最大化。所有任务智能体在系统中共享相同的奖励，以此鼓励智能体之间合作探索环境。

真实的环境状态 s_t 包括 VEC 线程的状态和所有任务智能体的状态。但是，单个任务智能体不能知道真实的全局状态，只能通过观测函数 O 观测局部状态。任务智能体 k 的观测空间包括线程的计算资源 f_m，线程处理时延 $T_{m,k}$，任务数据大小 d_k 和时延约束 τ_k。所以，任务智能体 k 的观测空间可以表示为

$$S_t^{(k)}=\{f_m,T_{m,k},d_k,\tau_k\} \tag{3-1}$$

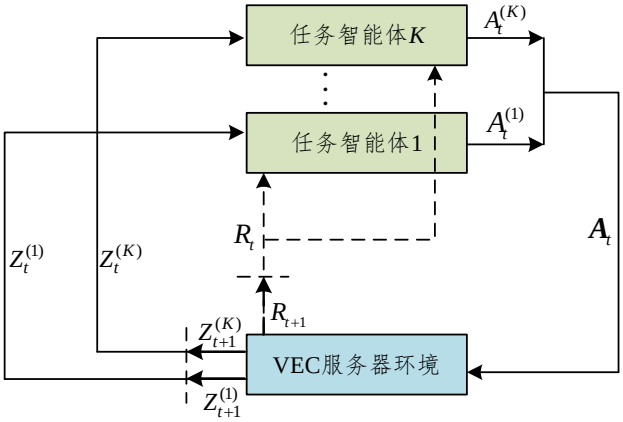

图 3-1　任务智能体和 VEC 服务器环境交互过程

Fig.3-1　Task agent and VEC server environment interaction process

3.2.2　动作空间

基于 MADQN 任务调度策略的目的是在考虑当前 VEC 服务器状态信息下，任务 k 从 M 个服务器线程中选择最优的线程处理，以达到减少 VEC 任务处理总时延的目的。所以每个任务智能体 k 的动作就是对 M 个线程的选择，即调度变量 $x_{k,m}$。所以，动作空间 A_t 可以表示为

$$A_t=\{[x_{1,1},...,x_{1,m}],...,[x_{k,1},...,x_{k,m}],...,[x_{K,1},...,x_{K,m}]\} \tag{3-2}$$

3.2.3　奖励设计

在任务智能体与 VEC 服务器环境交互过程中，奖励函数是环境对任务智能体动作的反馈，该反馈可用于指导任务智能体学习。因此系统奖励需要与所求目标一致才能提升系统的性能。本文目标有两个：最小化线程的任务处理时延和 VEC 服务器的总时延，

同时在时间限制 τ_k 下提高计算任务的成功处理概率。为达到上述目标，本文将多智能体深度 Q 网络的系统奖励 R_t 设计为

$$R_t = \begin{cases} \min\limits_{m\in\mathcal{M}} T_m - \max\limits_{m\in\mathcal{M}} T_m, & \text{if } T_{k,m} \le \tau_k \\ \beta, & \text{otherwise} \end{cases} \tag{3-3}$$

其中 β 是未能在任务时延约束内完成的惩罚，是一个根据实验经验设置的常数。β 的设置可以使任务智能体在训练中学习到让任务在时延约束内完成，以获得更大奖励 R_t 的策略。由公式（3-3）可知奖励 R_t 为负值，这是为了让智能体寻找最大奖励和本文优化目标最小化 VEC 服务器总时延相匹配，以提升系统的性能。

多智能体深度 Q 网络算法最终目标是找到最优策略 π_*，最大化期望折扣累计奖励 G_t，G_t 表示为

$$G_t = \mathbb{E}_\pi \left\{ \sum_{k=0}^{\infty} \lambda^k R_{t+k+1} \right\}, 0 \le \lambda \le 1. \tag{3-4}$$

其中 λ 是折扣因子，表示下一个时隙奖励对当前时隙奖励的影响系数。

3.3 MADQN 任务调度算法

MADQN 算法结构如图 3-2 所示，在算法学习过程中，每个任务智能体构建评估网络来生成调度策略。网络的输入是服务器状态空间 s，输出是对应状态的调度价值函数 $Q(s,a;\omega)$，其中 ω 为评估网络的权重参数。每个任务智能体构建目标网络用于表示下一个状态的调度价值函数 $Q(s',a';\omega')$，作为评估网络的目标值，其中 ω' 为目标网络的权重参数。

本文每个任务智能体的评估网络和目标网络都由一个输入层，三个隐藏层和一个输出层组成。每一层都由多个神经元组成，并和下一层的神经元连接。互相连接的神经元之间有相关的权重参数，不同的权重参数表示了变量的重要性程度，较大的权重参数对输出的贡献更大。本文中，输入层的神经元个数是状态空间 s 的长度。三个隐藏层的神经元个数分别是 256,128 和 64。输出层的神经元个数为服务器线程数 M，当智能体从输出层选择最大调度价值函数 Q 时，也表示了对服务器线程的选择。

训练过程中，用损失函数 $loss$ 来评估网络的准确性。训练的目标就是通过调整权重参数，使损失函数最小化。在权重参数更新过程中，只更新评估网络的权重 ω，目标网络的权重 ω' 更新是在完成一定回合数后，从评估网络复制更新 $\omega' \leftarrow \omega$。这样，评估网络权重更新针对的目标在一段时间内就是一个固定目标，增加了学习的稳定性。每一个任务智能体都有独立的评估网络和目标网络，也就是独立的 DQN，而经验回放是公用的，以鼓励任务智能体合作探索环境。

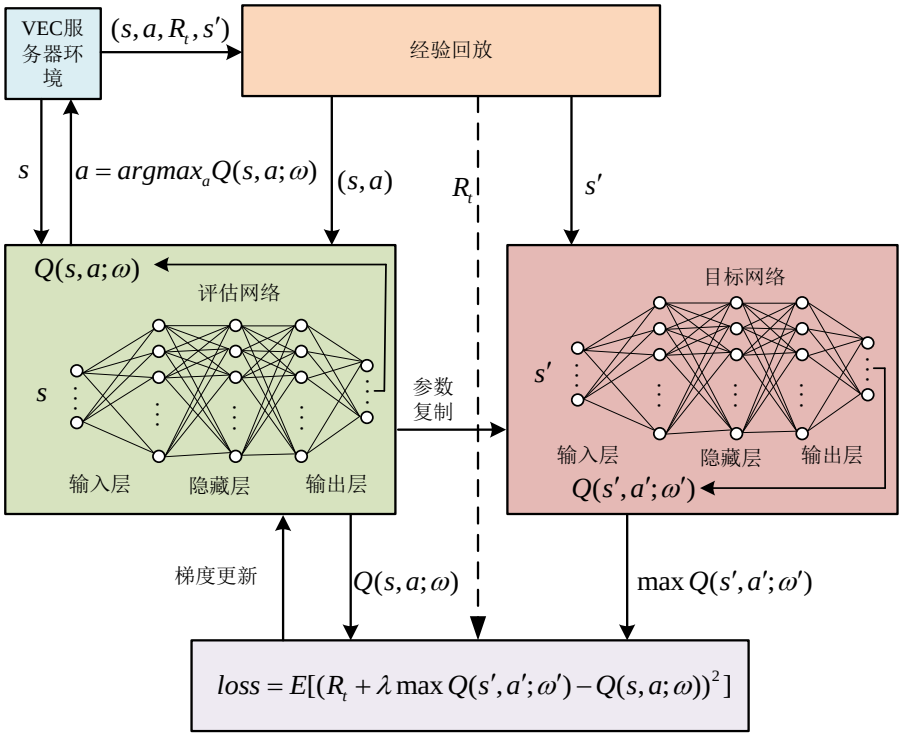

图 3-2　MADQN 算法结构

Fig.3-2　MADQN algorithm structure

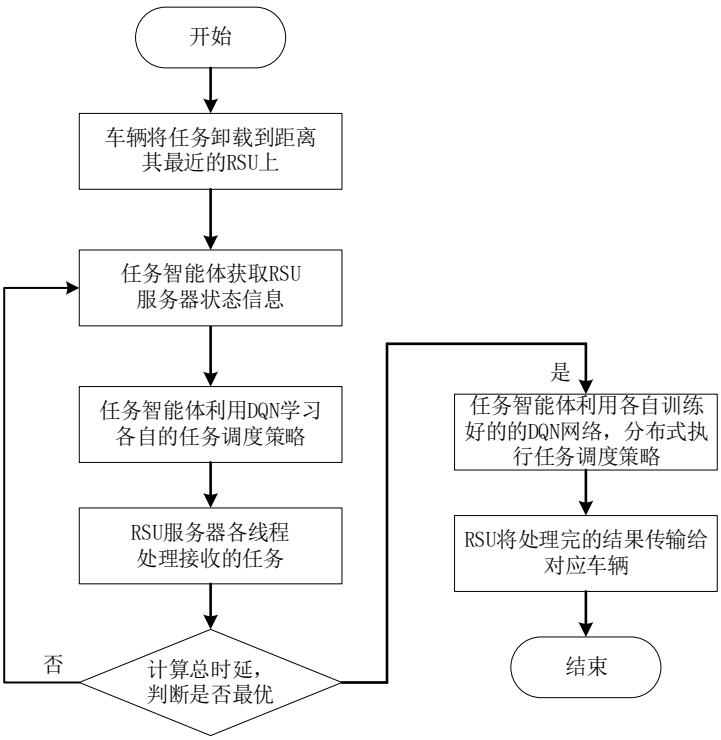

图 3-3　MADQN 训练流程

Fig.3-3　MADQN training process

3.3.1 训练过程

MADQN 训练流程如图 3-3 所示，在任务智能体的学习过程中，每一个回合从初始化 VEC 服务器环境状态和任务开始，直到任务调度完成结束。首先初始化评估网络的权重参数 ω，目标网络的权重参数 $\omega' \leftarrow \omega$，VEC 服务器环境状态 s_t。之后任务智能体评估网络接收环境状态 s_t，输出所有动作价值函数 $Q(s_t, a_t; \omega)$，$Q(s_t, a_t; \omega)$ 表示为

$$Q(s_t, a_t; \omega) = R_t + \lambda \mathrm{E}[Q(s_{t+1}, a_{t+1}; \omega)] \tag{3-5}$$

R_t 是线程处理即时奖励。然后任务智能体从所有动作价值函数 $Q(s_t, a_t; \omega)$ 中选择最大的 Q 值，执行最大 Q 值对应的动作 $a_t = argmax_a Q(s_t, a_t; \omega)$，环境进入下一个状态 s_{t+1}，并将经验 (s_t, a_t, R_t, s_{t+1}) 存入经验回放。

之后任务智能体从经验回放中均匀选取一批经验 $\mathcal{K}$，并将 s_{t+1} 作为目标网络的输入。然后目标网络输出状态 s_{t+1} 所有动作价值 $Q(s_{t+1}, a_{t+1}; \omega')$，再计算评估网络的目标值 U_t，U_t 表示为

$$U_t = R_t + \lambda \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}; \omega') \tag{3-6}$$

评估网络参数 ω 的更新是通过最小化损失函数 $loss$。损失函数 $loss$ 表示为

$$loss \triangleq \frac{1}{B} \sum_{\mathcal{K}} [U_t - Q(s_t, a_t; \omega)]^2 \tag{3-7}$$

其中 B 是采样样本大小。$loss$ 的梯度计算表示为

$$\nabla_w loss = \frac{1}{B} \sum_{\mathcal{K}} 2[U_t - Q(s_t, a_t; \omega)] \nabla Q(s_t, a_t; \omega) \tag{3-8}$$

评估网络参数 ω 更新公式 ω 如下：

$$\omega \leftarrow \omega + \eta \frac{1}{B} \sum_{\mathcal{K}} 2[U_t - Q(s_t, a_t; \omega)] \nabla Q(s_t, a_t; \omega) \tag{3-9}$$

其中 η 是评估网络的学习率。评估网络的权重参数 ω 更新完，再把 $s \leftarrow s_{t+1}$。重复训练直到所有任务调度完成，再结束这个回合。之后每进行一定回合数，更新目标网络的权重参数 $\omega' \leftarrow \omega$。MADQN 算法如算法 1 所示。在算法 1 中，我们用 $\varepsilon-$贪婪策略来探索状态和动作空间。这就意味着以 ε 的概率选择随机动作，以 $1-\varepsilon$ 的概率选择 Q 值最大的动作。ε 是一个全局变量，它随着训练次数的参加不断减少，最后趋于一个很小的定值。

3.3.2 分布式实现

在分布式实施阶段，每个时间步 t，任务智能体接收 VEC 环境状态，并根据其训练的 Q 网络选择奖励值最大的动作。然后，VEC 服务器的每个线程开始处理接收到的任务。值得注意的是，算法 1 中的计算密集型训练过程可以离线执行。当环境特征发生重大变化时，需要根据环境动态和系统性能要求更新训练的 Q 网络。多智能体通过分布式实施，使智能体可以通过智能体之间的协助共同解决问题。

表 3-1　基于 MADQN 的任务调度算法流程

算法 1 基于 MADQN 的任务调度算法
输入：评估网络及其参数 ω，目标网络及其参数 ω'，初始状态 s_t。
输出：任务调度策略。
1. **for** each episode **do**
2.　**for** each step *t* **do**
3.　　**for** each agent **do**
4.　　评估网络接收 s_t，以 ε 概率随机选择动作 a_t，以 $1-\varepsilon$ 概率选取 $a_t = argmax_a Q(s_t, a_t; \omega)$，$\varepsilon$ 随着训练回合不断减小；
5.　　end for
6.　　执行动作 $\boldsymbol{a}_t$，得到环境奖励 R_t，环境更新到下一个状态 s_{t+1}；
7.　　**for** each agent **do**
8.　　将经验 (s_t, a_t, R_t, s_{t+1}) 存入经验回放中
9.　　end for
11.　　**for** each agent **do**
12.　　从经验回放中选取一批经验 $\mathcal{K}$
13.　　计算目标值 U_t
14.　　更新 ω 以减少 $loss$
15.　　一定回合数后更新目标网络的权重 $\omega' \leftarrow \omega$
16.　　end for
17.　end for
18. end for

3.4　实验结果与分析

本文的实验环境为 Python 3.6，Tensorflow 1.12.0，Numpy 1.19.5。显卡为 NVIDIA GeForce RTX 3060，处理器 CPU 为 8 核，内存为 32G。VEC 环境参数和神经网络的参数设置如表 3-2 所示。其中任务数据大小采用均匀分布可以更好的得到训练数据。

图 3-4 显示了随着训练回合数的增加而增加的训练奖励，以研究所提出的 MADQN 方法的收敛性。我们可以看到，每回合的训练奖励随着训练回合数的增多而提高。当回合数达到约 2500 时，性能逐渐收敛。为了提供安全的收敛保证，我们对每个代理的 Q 网络进行了 3000 集的训练。图 3-5 显示了任务处理总时延随着训练回合数的增加而减少，以研究所提出的 MADQN 方法的有效性。我们可以发现图 3-5 的收敛趋势和图 3-4 收敛趋势大致相同，不同的是收敛的结果。图 3-5 表现的任务处理总时延随着训练回合数的增加而减少，最终收敛的趋势，证明了所提出训练算法的有效性。

为了验证本文所提出算法的性能，将选择以下两种算法进行比较：

（1）SADQN：Single Agent Deep Q-Networks，SADQN 算法是传统的深度强化学习算法，只训练一个 DQN 网络，所有任务都由一个 DQN 网络处理[61]。

表 3-2　实验参数设置

仿真参数	参数值
任务数 k	10
任务数据大小 d_k	U（5,50）MB，U（50,200）MB[59]
任务时延约束 τ_k	10~50ms[59]
服务器线程数 M	4
服务器线程总计算资源	8GHz
服务器线程计算资源 φ	[0.5,1.5,2.5,3.5]GHz
线程等待时间 $\mathcal{T}$	1~3ms
为任务配置的计算资源 e	0.001GHz/MB[60]
神经网络层数和隐藏层数	5,3
隐藏层神经元数	[256,128,64]
神经网络激活函数	Relu
梯度下降优化算法	RMSProp
神经网络学习率 η	0.001[39]
折扣因子 λ	0.99
常数 β	-1
目标网络参数更新回合数	100

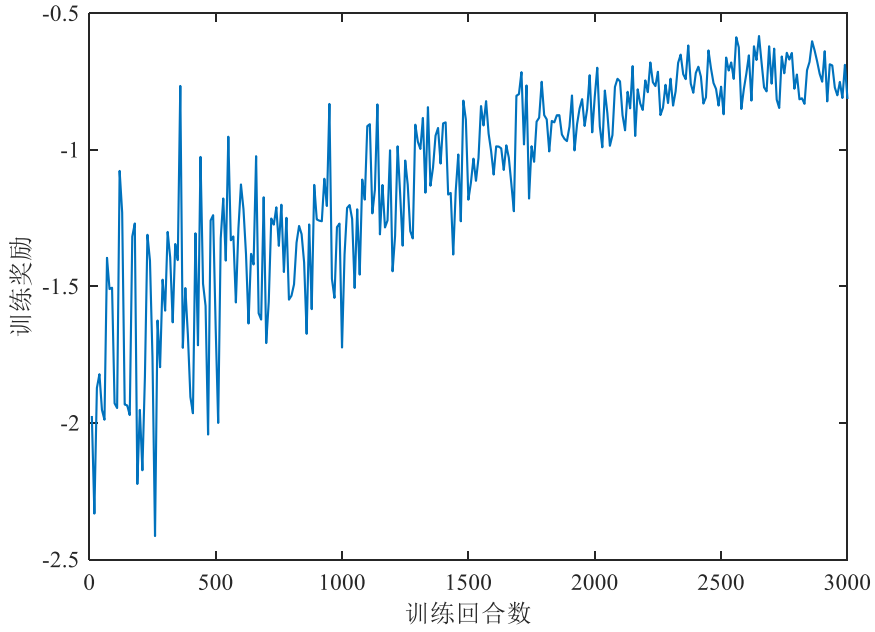

图 3-4　MADQN 训练奖励

Fig.3-4　MADQN training reward

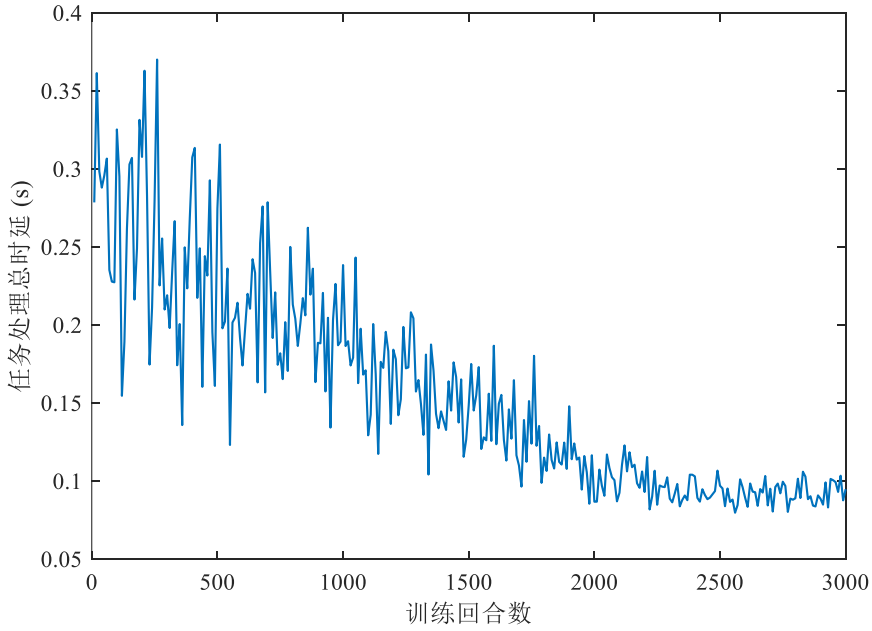

图 3-5　MADQN 任务处理总时延

Fig.3-5　MADQN total task processing delay

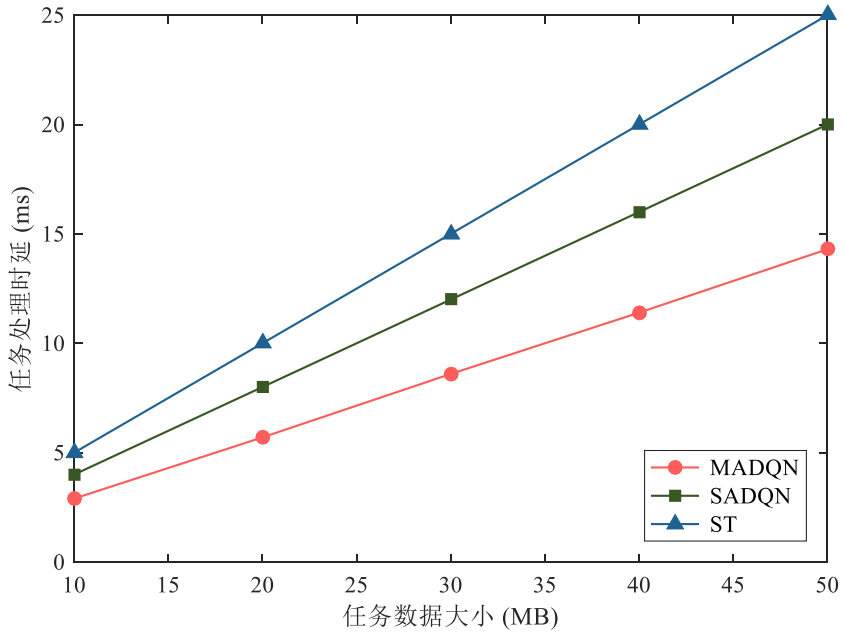

图 3-6　任务数据大小和任务处理时延的关系

Fig.3-6　Relationship between task data size and task processing delay

（2）ST：Shortest Time，ST 算法采用最短线程时间分配任务。线程任务处理时间最小的线程优先分配任务，线程任务处理时间最大的线程最后分配任务[62]。

图 3-6 所示为任务数据大小对 MADQN，SADQN 和 ST 三种调度算法任务处理时延的影响。由图 3-6 可以看出，任务处理时延和任务数据大小成正比。随着任务数据的

不断增加，任务处理时延也会增大，但无论在哪种数据大小下，MADQN 算法的任务处理时延都是最小的。比较三种算法的任务计算时延可知，MADQN 比 SADQN 和 ST 的任务处理时延分别减少了约 28%和 40%。

图 3-7 所示为任务数据大小对三种不同调度算法任务处理成功概率的影响。由图 3-7 可以看出，任务处理成功概率随着任务数据大小的增加而减少。分析图 3-7 可以看出，当任务数据大小超过 30MB，任务处理成功概率的开始明显下降。这是由于任务数据的增大，所需的计算资源也越来越多，而服务器线程的计算资源是不变，当这些任务调度到计算资源不足的线程进行处理，任务将超时完成。ST 算法尤为明显，是因为 ST 算法不能随着任务信息和服务器状态信息的改变而做出相应的决策。比较三种算法的任务处理成功概率可知，MADQN 比 SADQN 和 ST 分别提高了 5%~33%和 25%~100%。

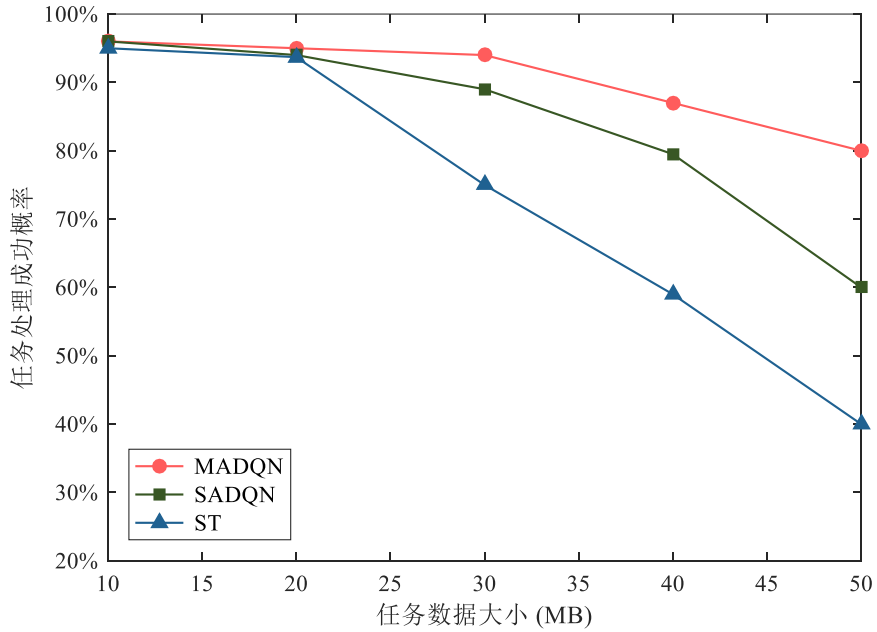

图 3-7　任务数据大小和任务处理成功概率的关系

Fig.3-7　The relationship between task data size and task processing success probability

为进一步研究了所提出算法对 VEC 服务器过载状态的计算性能。在图 3-8 中，本文模拟了均匀分布 U（50，200）MB 的计算任务，并不断增加任务数量。由图 3-8 可以看出，与 SADQN 和 ST 算法相比，MADQN 算法的总处理延迟显著减少。特别是，随着计算任务的增加，性能增益不断提高。这意味着所提出的基于 MADQN 的算法适用于过载的 VEC 场景，可以有效地解决车辆网络中交通高峰时段的计算问题。

图 3-9 所示为不同任务数约束下三种算法的任务处理成功概率对比。由图 3-9 可以看出，随着任务数的不断增加，三种算法的任务处理成功率不断降低。这是由于 VEC 服务器的计算资源 F 保持不变，而任务数的不断增加使所需的计算资源也不断增加，最终

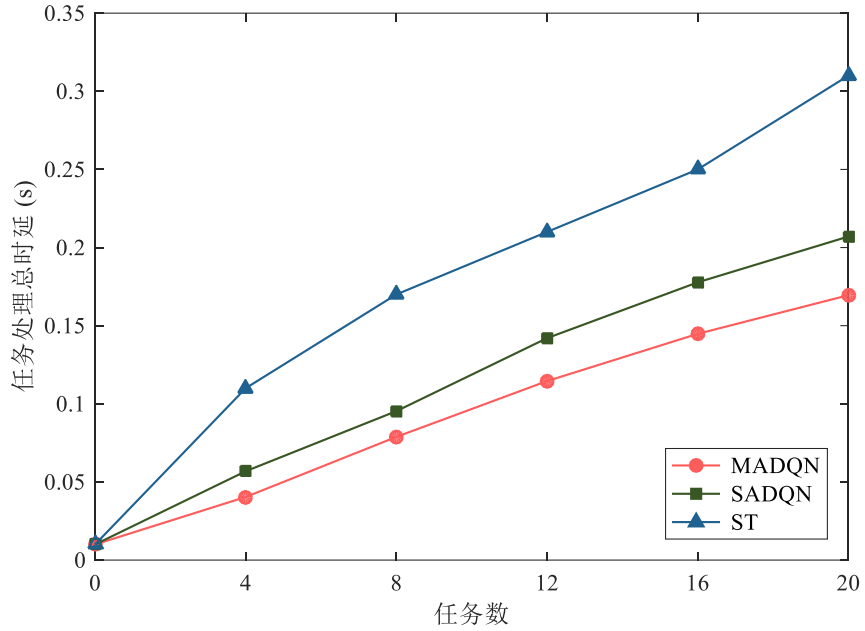

图 3-8　任务处理总时延与任务数的关系

Fig.3-8　The relationship between the total task processing delay and the number of tasks

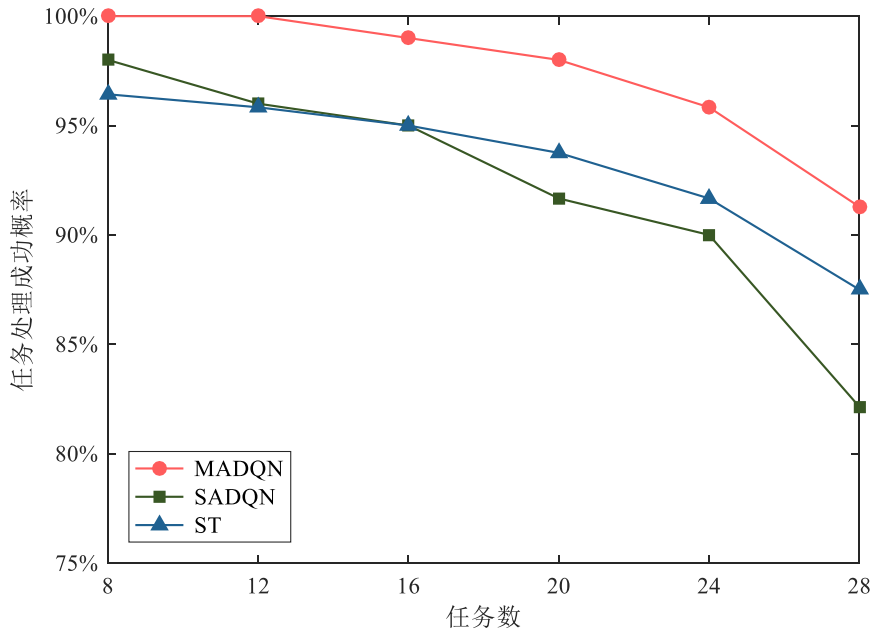

图 3-9　任务处理成功概率和任务数的关系

Fig.3-9　The relationship between the success probability of task processing and the number of tasks

使服务器超负荷运行，导致任务处理成功概率降低。分析三种算法曲线可以发现，当任务数在 12 以下时，MADQN 算法的成功率可以维持在 100%；当任务数超过 20 时，三种算法的任务处理成功率开始快速降低。所以，为保障任务的服务质量，卸载到 VEC 服

务器的任务数不宜超过 20，最优卸载任务数应控制在 12 以下。比较三种算法的曲线，MADQN 的任务处理成功率比 SADQN 和 ST 分别提高了 2%~9%和 5%。

为进一步体现 MADQN 算法对动态车联网边缘计算环境的适用性，本文设计了表 3-3 所示的三种任务类型，通过不同的任务来体现环境的变化。结合图 3-10 可以看出，当任务类型单一，如任务类型 1 和任务类型 2，SADQN 算法和 ST 算法任务处理总时延相差不大，MADQN 算法任务处理总时延相比它们减少约 20%。而当任务类型多样，如任务类型 3，MADQN 算法任务处理总时延比其他两种算法分别减少了 30%和 60%。与单一类型任务相比，MADQN 算法任务处理总时延比 ST 算法减少幅度提高了 3 倍。由此表明，MADQN 调度算法在动态车联网环境下高效性和适用性。

表 3-3 三种任务类型

任务类型	任务类型组成
任务类型 1	10 个输入数据为 5MB 的任务
任务类型 2	10 个输入数据为 50MB 的任务
任务类型 3	5 个输入数据为 5MB 的任务 和 5 个输入数据为 50MB 的任务

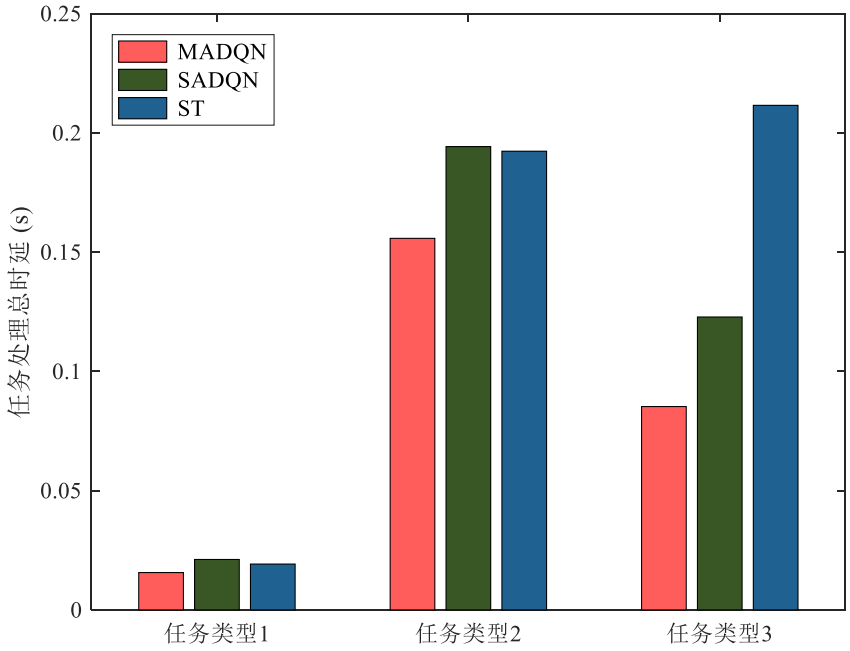

图 3-10 任务类型和任务处理总时延的关系

Fig.3-10　The relationship between task type and total task processing delay

为体现不同线程计算资源分配和任务处理总时延的关系。本文在保存 F=8GHz/s 不变的情况下，设计了表 3-4 所示的四种线程类型，其中任务类型相同。结合图 3-11 可知，线程类型 A 的任务处理总时延最短，线程类型 D 的服务器任务处理时延最长，线程类型 A 比线程类型 D 的任务处理总时延减少了约 44%。通过比较线程类型 A 和 D，表明

线程计算资源分配的越均匀，服务器任务处理时延越长。这也说明了面对任务的多样性，设置具有差异线程计算资源的必要性，为之后 VEC 服务器线程计算资源部署提供了参考。

表 3-4 线程类型和线程计算资源分配

线程类型	线程计算资源分配
线程类型 A	0.5GHz/s，1.5GHz/s，2.5GHz/s，3.5GHz/s
线程类型 B	1GHz/s，1.5GHz/s，2.5GHz/s，3GHz/s
线程类型 C	1.5GHz/s，2GHz/s，2GHz/s，2.5GHz/s
线程类型 D	2GHz/s，2GHz/s，2GHz/s，2GHz/s

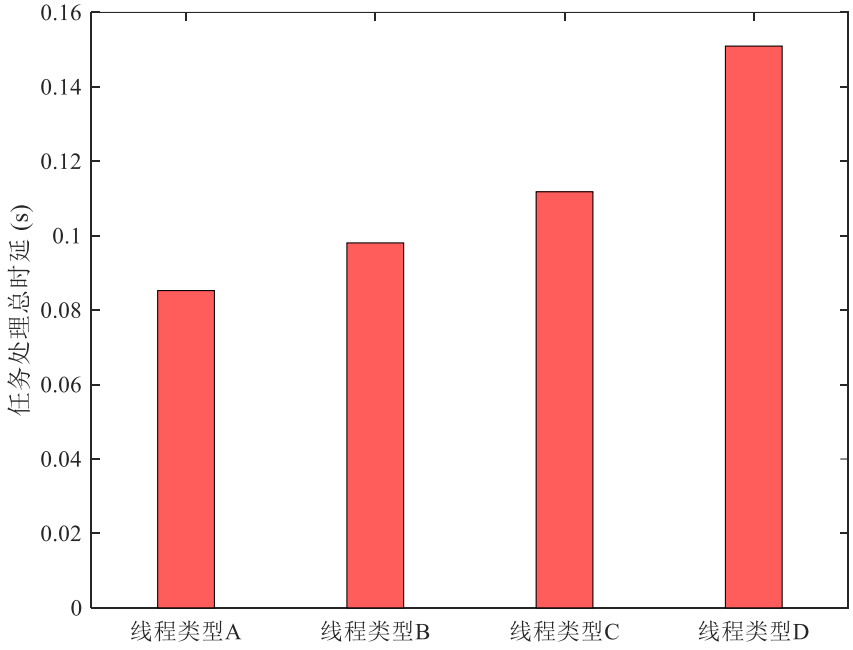

图 3-11 任务处理总时延与线程类型的关系

Fig.3-11 The relationship between Total Task Processing Delay and Thread Type

3.5 本章小结

为了解决 VEC 网络交通高峰时段的任务计算问题，本文提出了基于 MADQN 任务调度算法，以减少 VEC 服务器任务处理时延，提高 VEC 车辆的服务质量。针对 NP-hard 的优化问题，将优化问题建模成多智能体强化学习问题，并用 MADQN 算法求解任务调度策略。为提高 MADQN 算法的效率，设计了以线程时间差为奖励的奖励函数，并提出了任务智能体集中训练和分布式运行的机制。通过和其他两种算法的对比实验表明，本文所提算法能拥有更低的任务处理时延和更高的任务处理成功率，最终实现 VEC 低时延、高可靠的数据通信。

第四章　基于多智能体双重深度 Q 网络的车辆边缘计算任务调度

4.1　引言

在上一章中，本文对 VEC 密集型任务场景中的计算任务调度进行研究，提出了基于多智能体深度 Q 网络的任务调度算法。尽管 DQN 算法为 VEC 密集型任务场景提供了一个任务调度策略，但是 DQN 算法存在对 Q 值偏高估计的问题，会导致任务超时完成的概率增加。具体来说，Q 值偏高估计会导致任务都在计算资源较多的线程进行计算，使该线程处于饱和状态，而其他线程的计算资源没得到充分利用。基于此，对 DQN 算法的不足进行改进具有积极的意义。针对 DQN 算法 Q 值偏高估计的问题，本节提出一种基于 DDQN 的多智能体双重深度 Q 网络（Multi Agent Double Deep Q-Networks, MADDQN）任务调度算法来进行改进。

4.2　多智能体双重深度 Q 网络算法

DDQN 是一种针对传统 DQN 的改进算法，通过使用双重 Q 网络结构，能够有效地减少过估计（overestimation）问题，从而提高算法的收敛速度和性能。具体而言，过估计是指目标网络在估计值函数时会比真实值函数要大，其根源主要在于 DQN 中目标网络的最大化操作 $R_t + \lambda \max_{a'} Q(s', a'; \omega')$，其中的目标网络选择 Q 值最大操作使得估计的目标值函数比真实值函数更大。这是因为在真实情况下，选择 Q 值的概率随机的，而在 DQN 的目标网络中，每次都是选择 Q 值最大的，这个操作就会使 DQN 存在过高估计的问题，使目标网络的 Q 值高于真实的 Q 值，从而使训练后的 Q 值偏高。为了解决值函数过估计的问题，提出了基于双重 Q 网络结构的 DDQN 优化算法，其定义是将动作的选择和动作的评估分别用不同的值函数来实现。如图 4-1 所示的 MADDQN 算法结构，在选择动作 a' 时，DDQN 是用评估网络并根据 $a' = \arg\max_{a'} Q(s', a'; \omega)$ 选择使 Q 值最大的动作。在计算目标 Q 时，用目标网络根据输入的 s' 和 a' 计算出目标 Q 值。本文对目标 Q 值的计算公式 U_t^{DDQN} 为：

$$U_t^{DDQN} = R_t + \lambda Q(s_{t+1}, \arg\max_{a_{t+1}} Q(S_{t+1}, a_{t+1}; \omega); \omega') \tag{4-1}$$

相应的损失函数 $loss$ 也发生改变，$loss$ 可以表示为：

$$loss = E[(R_t + \lambda Q(s', \arg\max_{a'} Q(s', a'; \omega); \omega') - Q(s, a; \omega))^2] \tag{4-2}$$

这种用双 Q 网络来计算目标 Q 值的结构，有效的解决了 DQN 过估计的缺陷，使 Q 值更接近真实值，减少了估计偏差损失函数 $loss$ 的值，提高了算法的收敛速度和性能。

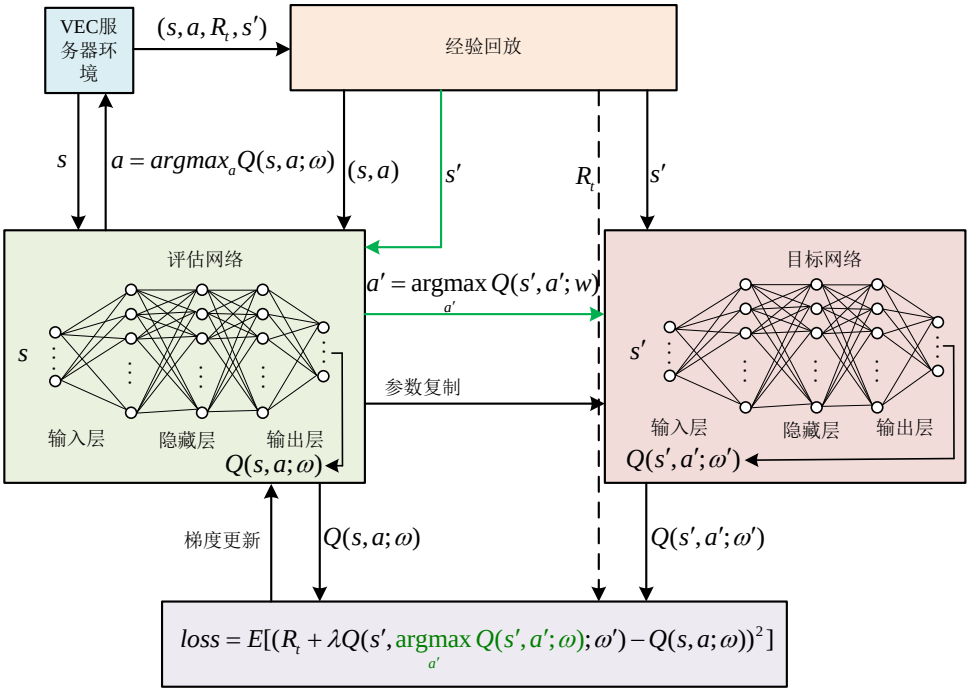

图 4-1　MADDQN 算法结构

Fig.4-1　MADDQN algorithm structure

表 4-1　MADDQN 算法流程

算法 2　Multi Agent Double Deep Q Network 算法

输入：初始化经验回放池、评估网络参数 ω 和目标网络参数 ω'

输出：训练后的网络和最优策略

重复执行以下步骤直到达到最大迭代次数或任务终止：

for each agent **do**：

1：根据当前状态 s，使用 $\varepsilon-\text{greedy}$ 策略选择动作 a

2：执行动作 a，获得新状态 s' 和即时奖励 R

3：将 (s,a,R,s') 存入经验回放池

end for

4：从经验回放池采样一批 (s,a,r,s')

for each agent **do**：

5：评估网络计算出 $Q(s,a;\omega)$

6：评估网络接收 s' 并输出 $a'=\underset{a'}{\operatorname{argmax}}\,Q(s',a';\omega)$

7：目标网络接收 s' 和 a' 并输出目标 $Q(s',a';\omega')$

8：计算损失函数，并通过梯度更新评估参数 ω

9：一定回合后从评估网络复制参数 ω 更新目标网络参数 ω'

end for

MADDQN 的算法流程如算法 2 所示。首先初始化经验回放池、评估网络参数 ω 和目标网络参数 ω'，之后每个智能体和环境交互，智能体接收输入环境状态 s，并使用 $\varepsilon-\text{greedy}$ 策略选择动作 a，执行动作 a，获得新状态 s' 和即时奖励 R，再将 (s,a,R,s') 存入经验回放池。当所有的智能体和环境结束一轮的交互，再从经验回放池中选取一批经验 (s,a,R,s')。接着每个智能体用评估网络计算出 $Q(s,a;\omega)$ 和输出 $a'=\underset{a'}{\text{argmax}}\,Q(s',a';\omega)$，之后再由目标网络计算出 $Q(s',a';\omega')$，最后计算损失函数并通过梯度更新评估参数 ω，一定回合后从评估网络复制参数 ω 更新目标网络参数 ω'。重复执行以上步骤直到达到最大迭代次数或任务终止，最终输出训练后的网络和最优策略。

4.3　实验结果与分析

本章的实验环境和第三章的实验环境一致，接下来比较分析 MADDQN 数据和 MADQN 数据。

首先，我们比较分析 MADDQN 和 MADQN 训练奖励的数据。图 4-2 展示了两种算法训练奖励随着训练回合数增加而变化的情况。由图 4-2 可以清楚看出，两种算法最终趋于收敛，这也表明了这两种算法的有效性。分析最终收敛的训练奖励大小，可以发现 MADDQN 算法的奖励高于 MADQN，具体数值提高约 30%。

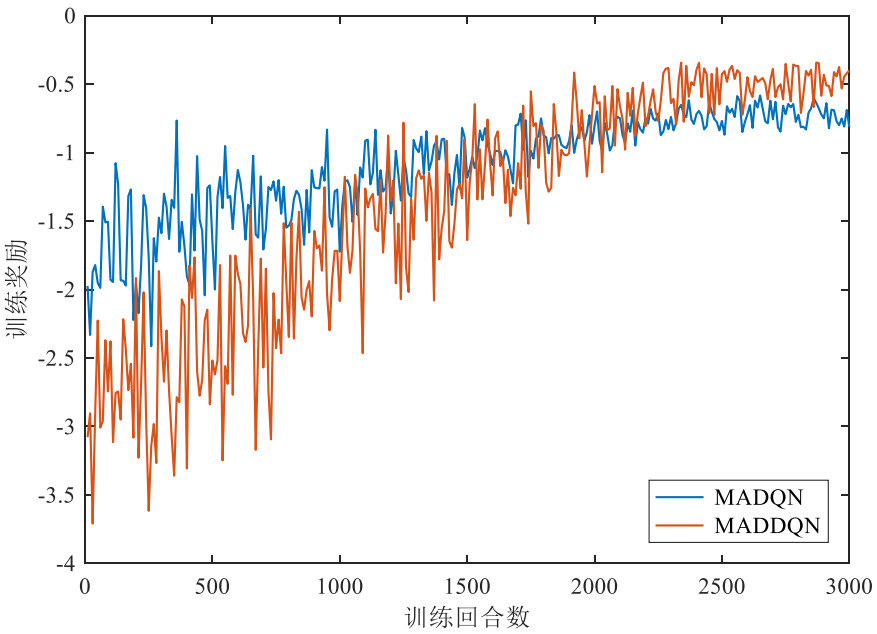

图 4-2　MADDQN 和 MADQN 训练奖励

Fig.4-2　MADDQN and MADQN training rewards

接着，我们分析 MADDQN 和 MADQN 任务处理总时延的比较。图 4-3 展示了两种算法任务处理总时延随训练回合数增加而变化的对比。由图中可以看出两种算法的任务

处理总时延最终都收敛，但 MADDQN 算法收敛的任务处理总时延比 MADQN 算法的更小，分析数据可知 MADDQN 算法收敛的任务处理总时延缩短了约 25%。结合图 4-2 的训练奖励的比较，我们可以得出 MADDQN 算法的整体性能比 MADQN 算法的整体性能提升约 20%~30%。

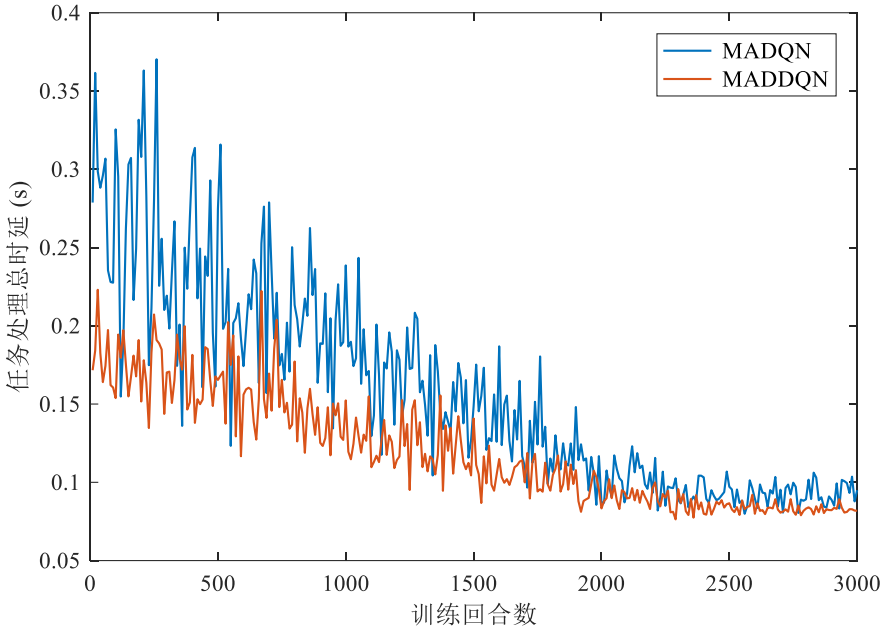

图 4-3　MADDQN 和 MADQN 任务处理总时延

Fig.4-3　MADDQN and MADQN total task processing delay

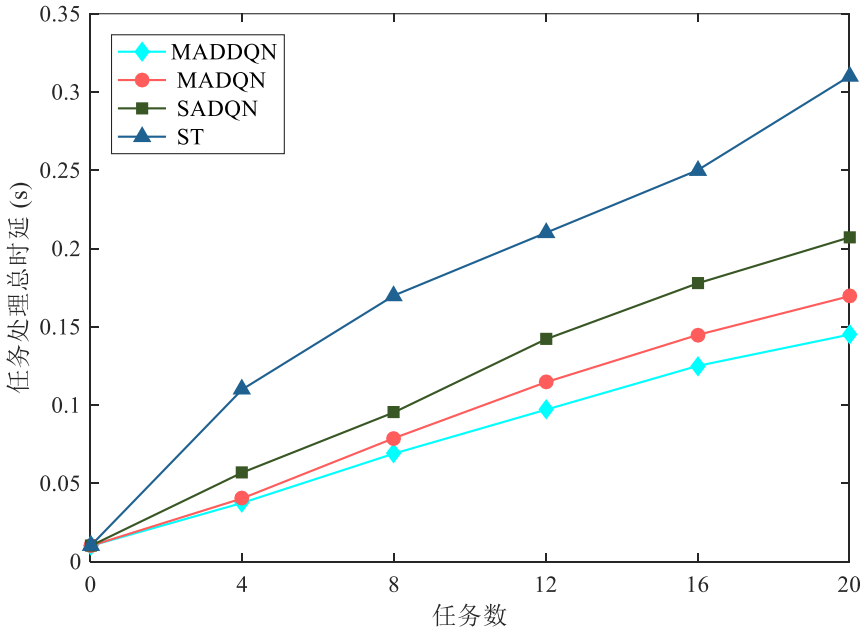

图 4-4　任务处理总时延与任务数的关系

Fig.4-4　The relationship between the total task processing delay and the number of tasks

下面我们将结和第三章的内容，将 MADDQN、MADQN、SADQN 和 ST 算法进行比较。首先，我们比较四种算法任务处理总时延和任务数的关系。图 4-4 展示了四种算法任务处理总时延随任务数增加而变化趋势。由图 4-4 可知，任务处理总时延和任务数成正比关系。仔细分析可知，MADDQN、MADQN、SADQN 三种深度强化学习算法在任务数增大后明显由于传统 ST 算法，这也表明了深度强化学习在面对复杂环境的处理效率更高。再比较三种深度强化学习算法，可以清楚发现 MADDQN 算法性能始终优于 MADQN 和 MADQN 算法，这表明了 DDQN 算法在改进目标 Q 值计算方式后，算法性能相较于 DQN 有明显提升。当任务数为 20 时，分析数据可知 MADDQN 任务处理总时延比 MADQN 任务处理总时延缩短了约 15%。

接着我们分析 MADDQN、MADQN、SADQN 和 ST 算法在任务处理成功概率和任务数的比较。图 4-5 展示了四种算法任务处理成功概率随任务数增加而变化的趋势。由图 4-5 可知，任务处理成功概率与任务数成反比关系。分析图 4-5 可知，SADQN 在任务数增加到 16 以后，任务处理成功概率开始处于四种算法中最低的位置，在任务数达到 24 之后，任务处理成功概率开始出现下坡式降低，这是因为单智能体在面对太复杂的环境时，智能体和环境的交互会明显减慢，任务处理成功概率也随之降低。而 MADDQN 和 MADQN 两种多智能体算法任务处理成功概率下降相较 SADQN 明显更少，是因为多智能体之间的协助，使多智能算法的稳定性得到提升。分析数据可知，在任务数为 28 时，MADDQN 算法任务处理成功概率约为 95%，MADQN 算法任务处理成功概率约为 91%，而 SADQN 算法任务处理成功概率约为 82%，相互比较可知 MADDQN 比 MADQN 高约 4%，比 SADQN 高约 13%。

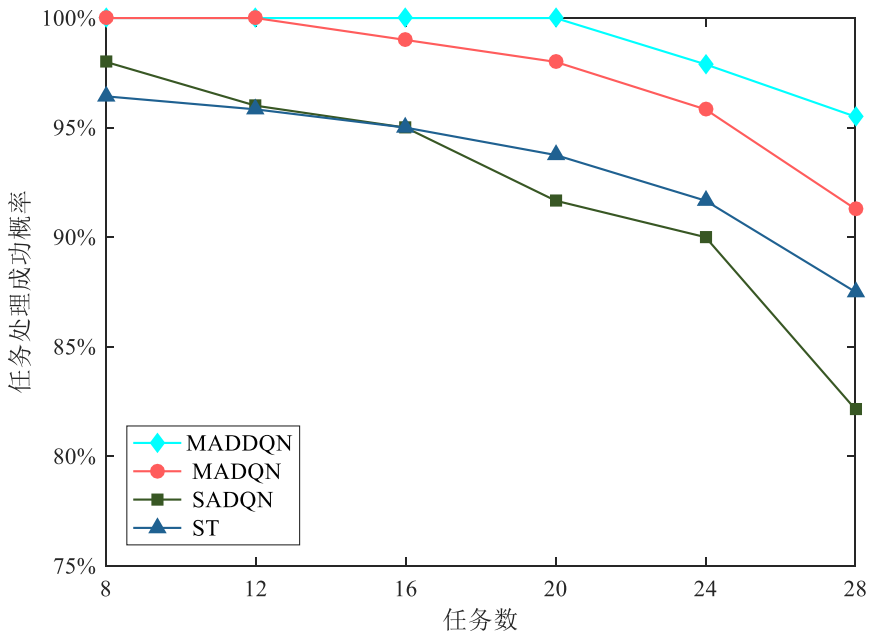

图 4-5　任务处理成功概率和任务数的关系

Fig.4-5　The relationship between the success probability of task processing and the number of tasks

最后，我们分析 MADDQN、MADQN、SADQN 和 ST 算法在任务类型和任务处理总时延关系比较。任务类型和第三章的表 3-3 的任务类型一致。图 4-6 展示了四种算法在不同任务类型下任务处理总时延的关系。由图 4-6 可以看出，在三种不同任务类型下，MADDQN 算法任务处理总时延总是优于 MADQN、SADQN 和 ST 算法。特别是在任务类型复杂的任务类型 3 情况下，MADDQN 算法任务处理总时延相较于 ST 算法任务处理总时延缩短了约 70%，相较于 SADQN 和 MADQN 缩短了约 45%和 25%。这也进一步表明了 MADDQN 算法比其他三种算法在处理复杂的环境更具优势。

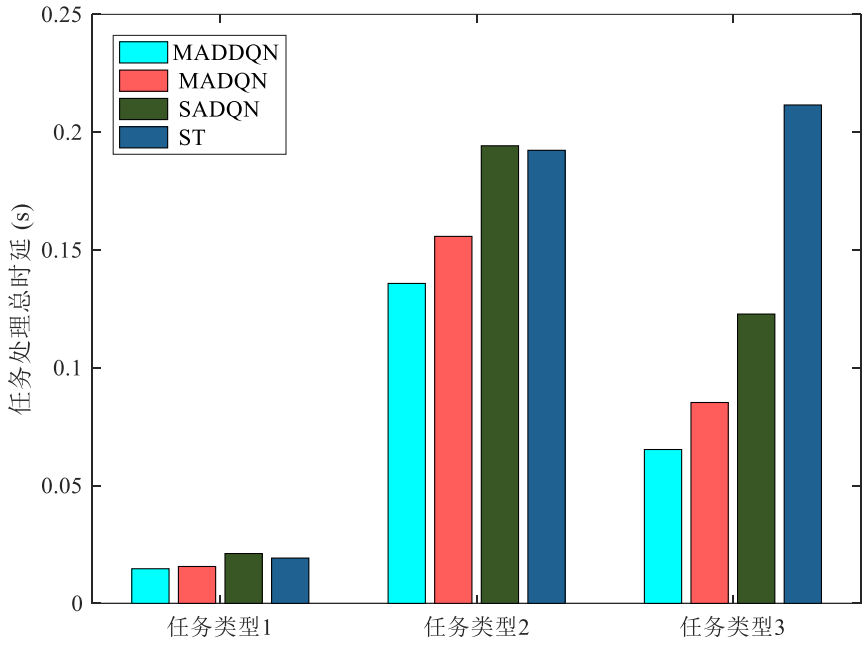

图 4-6　任务类型和任务处理总时延的关系

Fig.4-6　The relationship between task type and total task processing delay

4.4　本章小结

为了解决 DQN 算法对 Q 值偏高估计的问题，本文提出了基于 DDQN 的 MADDQN 任务调度算法，以优化算法对 Q 值的估计，从而进一步减少 VEC 服务器任务处理时延，提高 VEC 车辆的服务质量。简单来说，MADDQN 通过使用双重 Q 网络结构，将 Q 估计交由评估网络和目标网络共同完成，有效地减少过高估计问题，从而提高算法的收敛速度和性能。仿真结果表明 MADDQN 算法的整体性能比 MADQN 算法的整体性能提升约 20%~30%。在任务处理总时延、任务处理成功概率和复杂的任务类型下都优于其他算法。

第五章　总结与展望

5.1　工作总结

随着工业化和城市化的飞速发展，配备大量传感器和人机交互设备的智能汽车不断增加。然而车载终端计算资源和网络运营商部署 VEC 服务器的计算资源有限，在交通高峰期无法实现任务实时处理，为保证交通畅通和交通安全，实施高效的车辆边缘计算任务调度刻不容缓。本文从降低任务处理时延和提升任务处理效率角度出发，对任务密集型车联网中任务调度策略展开研究。本文的主要研究内容总结如下：

（1）针对 VEC 密集型任务的线程调度场景，提出基于 VEC 服务器的任务处理框架，并将多任务调度多线程的场景建模为多智能体强化学习问题，再用 MADQN 算法求解任务最优调度策略，以提升任务处理效率和减少任务处理时延。为提升 MADQN 算法的效率，在算法训练阶段，提出了基于线程时间的奖励机制，使任务调度优化问题与任务处理时延优化问题相结合；在算法执行阶段，提出分布式运行方式，以减少任务处理时延。仿真结果表明，本文提出的 MADQN 算法能够获得比现有基准算法更低的系统时延，并在密集型任务场景下拥有相对稳定的性能增益。

（2）针对 DQN 算法 Q 值偏高估计的问题，本文提出了一种基于 DDQN 的多智能体双重深度 Q 网络 MADDQN 任务调度算法来进行改进。MADDQN 首先由评估网络选择使 Q 值最大的动作，再由目标网络根据输入的状态和动作计算目标 Q 值，避免了只用目标网络导致的 Q 值偏高估计的问题。仿真结果表明，MADDQN 可以进一步提高任务处理效率。

5.2　未来展望

针对 VEC 任务密集型的任务调度场景，本文主要考虑任务处理时延，运用多智能深度强化学习寻找最优任务调度策略。但是对 VEC 任务调度场景仍有很多问题待解决，未来仍可对 VEC 任务调度场景的任务处理效率进行下一步的探索：

（1）本文第三章考虑了 VEC 任务调度场景中的边缘服务器进行任务处理，并未结合基站间合作处理任务。因此，下一步的研究可以考虑加入基站间合作处理任务模式，结合功率，频谱等约束条件，进行联合优化，降低系统能耗，进一步提高任务处理效率。

（2）本文第四章考虑了 DDQN 对 DQN 偏高估计的优化，并未对 DDQN 和其它深度强化学习算法进一步对比。因此，下一步的研究可以考虑和 Dueling DQN，Double Dueling Deep Q Network 等深度强化学习算法进行比较。

参考文献

[1] Kim T, Min H, Choi E, et al. Optimal job partitioning and allocation for vehicular cloud computing[J]. Future Generation Computer Systems, 2020, 108: 82–96.

[2] 前瞻产业研究院，《中国车联网行业市场前瞻与投资战略规划分析报告》.

[3] Guo H, Zhang J, Liu J. FiWi-Enhanced Vehicular Edge Computing Networks: Collaborative Task Offloading[J]. IEEE Vehicular Technology Magazine, 2019, 14(1): 45–53.

[4] Shi W, Cao J, Zhang Q, et al. Edge Computing: Vision and Challenges[J]. IEEE Internet of Things Journal, 2016, 3(5): 637–646.

[5] Ji L, Guo S. Energy-Efficient Cooperative Resource Allocation in Wireless Powered Mobile Edge Computing[J]. IEEE Internet of Things Journal, 2019, 6(3): 4744–4754.

[6] Kumar N, Zeadally S, Rodrigues J J P C. Vehicular delay-tolerant networks for smart grid data management using mobile edge computing[J]. IEEE Communications Magazine, 2016, 54(10): 60–66.

[7] Mao Y, You C, Zhang J, et al. A Survey on Mobile Edge Computing: The Communication Perspective[J]. IEEE Communications Surveys & Tutorials, 2017, 19(4): 2322–2358.

[8] Dai Y, Xu D, Zhang K, et al. Deep Reinforcement Learning and Permissioned Blockchain for Content Caching in Vehicular Edge Computing and Networks[J]. IEEE Transactions on Vehicular Technology, 2020, 69(4): 4312–4324.

[9] Liu J, Ahmed M, Mirza M A, et al. RL/DRL Meets Vehicular Task Offloading Using Edge and Vehicular Cloudlet: A Survey[J]. IEEE Internet of Things Journal, 2022, 9(11): 8315–8338.

[10] Wang Y, Lang P, Tian D, et al. A Game-Based Computation Offloading Method in Vehicular Multiaccess Edge Computing Networks[J]. IEEE Internet of Things Journal, 2020, 7(6): 4987–4996.

[11] 刘可欣, 陈桂芬. 基于移动边缘计算的车联网缓存策略研究[J]. 计算机应用研究, 2021, 38(03): 851-854+870.

[12] 孙嘉楠. 边缘计算环境下车联网任务卸载与数据分发技术研究[D]. 北京交通大学, 2020.

[13] Xu X, Xue Y, Li X, et al. A Computation Offloading Method for Edge Computing With Vehicle-to-Everything[J]. IEEE Access, 2019, 7: 131068–131077.

[14] 许世琳. 车联网中基于深度强化学习的计算任务卸载策略研究[D]. 北京邮电大学, 2021.

[15] Zheng K, Meng H, Chatzimisios P, et al. An SMDP-Based Resource Allocation in Vehicular Cloud Computing Systems[J]. IEEE Transactions on Industrial Electronics, 2015, 62(12): 7920–7928.

[16] Zhang K, Mao Y, Leng S, et al. Optimal delay constrained offloading for vehicular edge computing networks[C]. 2017 IEEE International Conference on Communications (ICC). Paris, France: IEEE, 2017: 1–6.

[17] Qiao G, Leng S, Zhang K, et al. Collaborative Task Offloading in Vehicular Edge Multi-Access Networks[J]. IEEE Communications Magazine, 2018, 56(8): 48–54.

[18] Xu J, Chen L, Zhou P. Joint Service Caching and Task Offloading for Mobile Edge Computing in Dense Networks[C]. IEEE INFOCOM 2018 - IEEE Conference on Computer Communications, Honolulu, HI, USA: IEEE, 2018: 207-215.

[19] Tan L T, Hu R Q, Hanzo L. Twin-Timescale Artificial Intelligence Aided Mobility-Aware Edge Caching and Computing in Vehicular Networks[J]. IEEE Transactions on Vehicular Technology, 2019, 68(4): 3086–3099.

[20] Yang L, Zhang H, Li M, et al. Mobile Edge Computing Empowered Energy Efficient Task Offloading in 5G[J]. IEEE Transactions on Vehicular Technology, 2018, 67(7): 6398–6409.

[21] Liu J, Wan J, Zeng B, et al. A Scalable and Quick-Response Software Defined Vehicular Network Assisted by Mobile Edge Computing[J]. IEEE Communications Magazine, 2017, 55(7): 94–100.
[22] Liu Y, Wang S, Huang J, et al. A Computation Offloading Algorithm Based on Game Theory for Vehicular Edge Networks[C]. 2018 IEEE International Conference on Communications (ICC). Kansas City, MO: IEEE, 2018: 1–6.
[23] Wang Z, Zhong Z, Ni M. A semi-Markov decision process-based computation offloading strategy in vehicular networks[C]. 2017 IEEE 28th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC). Montreal, QC: IEEE, 2017: 1–6.
[24] Sun F, Hou F, Cheng N, et al. Cooperative Task Scheduling for Computation Offloading in Vehicular Cloud[J]. IEEE Transactions on Vehicular Technology, 2018, 67(11): 11049–11061.
[25] Sun Y, Guo X, Zhou S, et al. Learning-Based Task Offloading for Vehicular Cloud Computing Systems[C]. 2018 IEEE International Conference on Communications (ICC). Kansas City, MO, USA: IEEE, 2018: 1-7.
[26] Haydar A, Yılmaz Y. Deep Reinforcement Learning for Intelligent Transportation Systems: A Survey[J]. IEEE Transactions on Intelligent Transportation Systems, 2022, 23(1): 11–32.
[27] Tang F, Mao B, Kato N, et al. Comprehensive Survey on Machine Learning in Vehicular Network: Technology, Applications and Challenges[J]. IEEE Communications Surveys & Tutorials, 2021, 23(3): 2027–2057.
[28] Liu N, Li Z, Xu Z, et al. A Hierarchical Framework of Cloud Resource Allocation and Power Management Using Deep Reinforcement Learning[C]. 2017 IEEE 37th international conference on distributed computing systems (ICDCS). Atlanta, GA, USA: IEEE, 2017: 372-382.
[29] Zhan Y, Yao J, Guan H. Intelligent Cloud Resource Management with Deep Reinforcement Learning[J]. IEEE Cloud Computing, 2017, 4(6): 60–69.
[30] Cheng M, Li J, Nazarian S. DRL-cloud: Deep reinforcement learning-based resource provisioning and task scheduling for cloud service providers[C]. 2018 23rd Asia and South Pacific Design Automation Conference (ASP-DAC). Jeju: IEEE, 2018: 129–134.
[31] Karthiban K, Raj J S. An efficient green computing fair resource allocation in cloud computing using modified deep reinforcement learning algorithm[J]. Soft Computing, 2020, 24(19): 14933–14942.
[32] Zhan W, Luo C, Wang J, et al. Deep-Reinforcement-Learning-Based Offloading Scheduling for Vehicular Edge Computing[J]. IEEE Internet of Things Journal, 2020, 7(6): 5449–5465.
[33] Luo Q, Li C, Luan T H, et al. Collaborative Data Scheduling for Vehicular Edge Computing via Deep Reinforcement Learning[J]. IEEE Internet of Things Journal, 2020, 7(10): 9637–9650.
[34] 丁飞, 沙宇晨, 洪莹, 等. 智能网联汽车计算卸载与边缘缓存联合优化策略[J]. 系统仿真学报, : 1–11.
[35] Ke H, Wang J, Deng L, et al. Deep Reinforcement Learning-Based Adaptive Computation Offloading for MEC in Heterogeneous Vehicular Networks[J]. IEEE Transactions on Vehicular Technology, 2020, 69(7): 7916–7929.
[36] Sun F, Cheng N, Zhang S, et al. Reinforcement Learning Based Computation Migration for Vehicular Cloud Computing[C]. 2018 IEEE Global Communications Conference (GLOBECOM). Abu Dhabi, United Arab Emirates: IEEE, 2018: 1–6.
[37] He X, Lu H, Du M, et al. QoE-Based Task Offloading With Deep Reinforcement Learning in Edge-Enabled Internet of Vehicles[J]. IEEE Transactions on Intelligent Transportation Systems, 2021, 22(4): 2252–2261.
[38] 杜威, 丁世飞. 多智能体强化学习综述[J]. 计算机科学, 2019, 46(08): 1–8.

[39] Liang L, Ye H, Li G Y. Spectrum Sharing in Vehicular Networks Based on Multi-Agent Reinforcement Learning[J]. IEEE Journal on Selected Areas in Communications, 2019, 37(10): 2282–2292.
[40] 宁兆龙, 张凯源, 王小洁, 等. 基于多智能体元强化学习的车联网协同服务缓存和计算卸载[J]. 通信学报, 2021, 42(06): 118–130.
[41] Zhu X, Luo Y, Liu A, et al. Multiagent Deep Reinforcement Learning for Vehicular Computation Offloading in IoT[J]. IEEE Internet of Things Journal, 2021, 8(12): 9763–9773.
[42] Jordan M I, Mitchell T M. Machine learning: Trends, perspectives, and prospects[J]. Science, American Association for the Advancement of Science, 2015, 349(6245): 255–260.
[43] Kaelbling L P, Littman M L, Moore A W. Reinforcement learning: A survey[J]. Journal of artificial intelligence research, 1996, 4: 237–285.
[44] LeCun Y, Bengio Y, Hinton G. Deep learning[J]. nature, Nature Publishing Group, 2015, 521(7553): 436–444.
[45] Mnih V, Kavukcuoglu K, Silver D, et al. Human-level control through deep reinforcement learning[J]. nature, Nature Publishing Group, 2015, 518(7540): 529–533.
[46] 李波, 牛力, 彭紫艺, 等. 基于马尔科夫决策过程的车载边缘计算切换策略[J]. 计算机工程与科学, 2020, 42(05): 788-794.
[47] Xu S, Guo C, Li Z. NOMA Enabled Resource Allocation for Vehicle Platoon-Based Vehicular Networks[C]. 2019 IEEE Globecom Workshops (GC Wkshps). Waikoloa, HI, USA: IEEE, 2019: 1–6.
[48] Van Hasselt H, Guez A, Silver D. Deep Reinforcement Learning with Double Q-Learning[J]. Proceedings of the AAAI Conference on Artificial Intelligence, 2016, 30(1).
[49] Silva F L D, Costa A H R. A Survey on Transfer Learning for Multiagent Reinforcement Learning Systems[J]. Journal of Artificial Intelligence Research, 2019, 64: 645–703.
[50] Qiao G, Leng S, Zhang K, et al. Collaborative Task Offloading in Vehicular Edge Multi-Access Networks[J]. IEEE Communications Magazine, 2018, 56(8): 48–54.
[51] Zhao J, Sun X, Ma X, et al. Online Distributed Optimization for Energy-Efficient Computation Offloading in Air-Ground Integrated Networks[J]. IEEE Transactions on Vehicular Technology, 2022: 1–14.
[52] Liu Y, Peng M, Shou G, et al. Toward edge intelligence: Multiaccess edge computing for 5G and Internet of Things[J]. IEEE Internet of Things Journal, IEEE, 2020, 7(8): 6722–6747.
[53] Yang M. Research on vehicle automatic driving target perception technology based on improved MSRPN algorithm[J]. Journal of Computational and Cognitive Engineering, 2022, 1(3): 147-151.
[54] Yang L, Zhang H, Li M, et al. Mobile Edge Computing Empowered Energy Efficient Task Offloading in 5G[J]. IEEE Transactions on Vehicular Technology, 2018, 67(7): 6398–6409.
[55] Liu Y, Wang S, Zhao Q, et al. Dependency-Aware Task Scheduling in Vehicular Edge Computing[J]. IEEE Internet of Things Journal, 2020, 7(6): 4961–4971.
[56] Zhao J, Sun X, Li Q, et al. Edge Caching and Computation Management for Real-Time Internet of Vehicles: An Online and Distributed Approach[J]. IEEE Transactions on Intelligent Transportation Systems, 2021, 22(4): 2183–2197.
[57] Li H, Xu H, Zhou C, et al. Joint Optimization Strategy of Computation Offloading and Resource Allocation in Multi-Access Edge Computing Environment[J]. IEEE Transactions on Vehicular Technology, 2020, 69(9): 10214–10226.
[58] Qi F, Zhuo L, Xin C. Deep Reinforcement Learning Based Task Scheduling in Edge Computing Networks[C]. 2020 IEEE/CIC International Conference on Communications in China (ICCC). Chongqing, China: IEEE, 2020: 835–840.

[59] Dai Y, Zhang K, Maharjan S, et al. Edge Intelligence for Energy-Efficient Computation Offloading and Resource Allocation in 5G Beyond[J]. IEEE Transactions on Vehicular Technology, 2020, 69(10): 12175–12186.

[60] Qi Q, Zhang L, Wang J, et al. Scalable Parallel Task Scheduling for Autonomous Driving Using Multi-Task Deep Reinforcement Learning[J]. IEEE Transactions on Vehicular Technology, 2020, 69(11): 13861–13874.

[61] Ye H, Li G Y, Juang B-H F. Deep Reinforcement Learning Based Resource Allocation for V2V Communications[J]. IEEE Transactions on Vehicular Technology, 2019, 68(4): 3163–3173.

[62] Singh K, Alam M, Kumar S. A Survey of Static Scheduling Algorithm for Distributed Computing System[J]. International Journal of Computer Applications, 2015, 129(2): 25–30.