学士论文版权使用授权书

本学士论文作者完全了解北京交通大学有关保留、使用学士论文的规定。特授权北京交通大学可以将学士论文的全部或部分内容编入有关数据库进行检索，提供阅览服务，并采用影印、缩印或扫描等复制手段保存、汇编以供查阅和借阅。

（保密的学位论文在解密后适用本授权说明）

学位论文作者签名: 指导教师签名:

签字日期: 年 月 日 签字日期: 年 月 日

学士论文诚信声明

本人声明所呈交的毕业论文（设计），题目基于云-边协同的边缘 LLM Agent 长任务处方法的设计与实现是本人在指导教师的指导下，独立进行研究工作所取得的成果。尽我所知，除了文中特别加以标注和致谢中所罗列的内容以外，论文中不包含其他人已经发表或撰写过的研究成果，也不包含为获得北京交通大学或其他教育机构的学位或证书而使用过的材料。

申请学位论文与资料若有不实之处，本人承担一切相关责任。

本人签名：_____ 日期：_____

中文摘要

摘要: 面向资源受限边缘环境中大语言模型智能体长任务处理能力不足的问题，本文围绕上下文膨胀、阶段经验积累不足和纯本地执行下知识复用受限等瓶颈，研究云边协同条件下的长任务处理方法，设计并实现了一种以边缘轻量执行、云端经验整理和反馈回流为核心的云边协同长任务处理方法。

本文的主要工作包括四个方面：一是构建边缘在线执行与云端知识管理相结合的方法架构，明确边缘侧低时延交互和云端侧经验整理的职责分工；二是设计子目标驱动的边缘执行机制和分层记忆组织方式，通过当前阶段细粒度保留与历史阶段摘要化维护控制工作记忆规模；三是构建云端短期缓存与长期知识库，将阶段执行结果转化为可复用的局部反馈与全局指导；四是实现长任务处理方法原型，并补充可视化监测模块，为协同链路观察和实验分析提供支撑。

实验在 PDDL 规划任务和 Jericho 文本交互任务上开展。AgentBoard 是面向大语言模型智能体的综合评测框架，本文参考其任务环境及成功率、任务推进率和动作落地准确率等任务完成类指标；HiAgent 是基于 AgentBoard 组织长任务执行与上下文压缩分析的智能体框架，本文参考其边缘长任务执行组织及平均执行步数、上下文消耗等执行开销类指标。本文设置两个递进式实验场景：场景一以 HiAgent 式本地弱模型执行设置为对照，验证云端经验回流机制对弱模型边缘节点的直接增益；场景二以场景一中的弱模型边缘节点为比较对象，验证强模型节点经验进入云端知识库后的增量作用。实验结果表明，相较本地弱模型对照，场景一中任务成功率提升 6.50 个百分点，任务推进率提升 10.12 个百分点，上下文消耗降低 27.88%；相较场景一，场景二在引入强节点经验后任务成功率进一步提升 1.00 个百分点，任务推进率进一步提升 1.79 个百分点，上下文消耗进一步降低 7.73%。消融实验进一步表明，云端经验回流机制是弱模型获得增益的关键来源，强节点经验能够提供额外但受限的补充作用。

本文通过边缘工作记忆控制、云端长期经验整理与反馈回流，缓解了边缘弱模型智能体在长任务处理中上下文受限和经验难以复用的问题，为资源受限边缘环境中的智能体持续执行、经验共享和协同增强提供方法支撑，并通过原型平台提供运行监测能力。

关键词：大语言模型智能体；云边协同；长任务处理；记忆管理；边缘智能

ABSTRACT

ABSTRACT:

For the insufficient long-horizon task processing capability of large language model agents in resource-constrained edge environments, this thesis focuses on context inflation, insufficient stage-level experience accumulation, and limited knowledge reuse under purely local execution. It studies long-horizon task processing under cloud-edge collaboration, and designs and implements a cloud-edge collaborative long-horizon task processing method centered on lightweight edge execution, cloud-side experience organization, and feedback delivery.

The main work consists of four aspects. First, this thesis constructs a method architecture that combines edge-side online execution with cloud-side knowledge management, clarifying the division between low-latency edge interaction and cloud-side experience organization. Second, it designs a subgoal-driven edge execution mechanism and a hierarchical memory organization method, controlling the working-memory scale through fine-grained current-stage records and summarized historical stages. Third, it constructs a cloud-side short-term cache and long-term knowledge base to transform stage execution results into reusable local feedback and global guidance. Fourth, it implements a prototype of the long-horizon task processing method and adds a visualization-based monitoring module to support observation of the collaboration process and experimental analysis.

Experiments are conducted on PDDL planning tasks and Jericho text-interaction tasks. AgentBoard is a comprehensive benchmark framework for large language model agents; this thesis uses its task environments and task-completion metrics, including success rate, progress rate, and grounding accuracy. HiAgent is an agent framework that organizes long-horizon execution and context-compression analysis based on AgentBoard; this thesis follows its edge-side execution organization and cost-related metrics, including average steps and context consumption. Two progressive scenarios are designed. Scenario 1 uses the HiAgent-style local weak-model execution setting as the baseline to evaluate the direct benefit of the cloud-side experience feedback mechanism for weak edge nodes. Scenario 2 uses the weak edge node in Scenario 1 as the comparison target to evaluate the incremental effect of adding strong-node experience to the cloud knowledge base. Results show that, compared with the local weak-model baseline, Scenario 1 improves the task success rate by 6.50 percentage points, improves the task progress rate by 10.12 percentage points, and reduces context consumption by 27.88%. Compared with Scenario 1, introducing strong-node experience in Scenario 2 further improves the task success

rate by 1.00 percentage point, improves the task progress rate by 1.79 percentage points, and reduces context consumption by 7.73%. Ablation experiments further show that the cloud-side experience feedback mechanism is the key source of the weak edge node's gains, while strong-node experience provides additional but bounded benefits.

Through edge working-memory control, cloud-side long-term experience organization, and feedback delivery, this thesis alleviates the problems of limited context and insufficient experience reuse for weak edge-side agents in long-horizon tasks, providing methodological support for continuous execution, experience sharing, and collaborative enhancement in resource-constrained edge environments, with the prototype platform providing runtime monitoring capability.

KEYWORDS: Large Language Model Agents; Cloud-Edge Collaboration; Long-Horizon Task Processing; Memory Management; Edge Intelligence

目　　录

1 引言

1.1 研究背景

近年来，以 ChatGPT、Gemini 等为代表的大语言模型持续演进，并在语言理解、知识组织与复杂推理等方面取得了显著进展[1,2]。国内关于大型语言模型原理与发展的综述指出，预训练、对齐、上下文学习和指令遵循等能力共同推动了大语言模型的发展[3]。相关研究进一步表明，大语言模型正逐步由静态文本生成工具演化为能够感知环境、维持上下文并输出可执行动作的智能体系统[4,5]。中文智能体综述也将基于大语言模型的智能体概括为包含感知、规划、记忆和行动等组件的系统形态[6]。工具调用和环境生成等研究进一步拓展了大语言模型智能体的动作空间与任务适应能力[7,8]。依托多轮交互和任务分解能力，这类系统已经在软件开发和机器人系统等场景中展现出较强的应用潜力[9,10]。面向对话推荐的用户模拟研究也说明，智能体方法可以服务于交互式系统评测[11]。社会行为模拟研究进一步说明，大语言模型智能体能够被用于开放式社会交互场景[12]。开放式具身智能体和浏览器辅助问答等工作也表明，大语言模型正在被用于更开放的连续交互环境[13,14]。具身智能相关综述强调，大模型驱动的具身系统需要将感知理解、任务规划和动作执行纳入统一闭环[15]。面向具身智能系统的综述进一步从感知、控制、系统架构和数据来源等角度总结了大模型在具身智能中的作用[16]。与传统大语言模型主要在静态、非交互环境中完成问题求解不同，智能体需要持续与外部环境交互，并在任务推进过程中不断根据新观测调整行为[5,17]。在这一过程中，记忆机制成为区分智能体与普通大语言模型的重要组成部分，它决定了系统如何积累知识、组织历史经验并检索与当前决策相关的信息[17,18]。围绕长期对话和长期记忆增强的研究进一步说明，记忆组织方式会直接影响智能体持续执行和经验复用能力[19,20]。

现有大语言模型智能体通常以观测、推理与动作交替展开的轨迹形式组织交互历史，并在推理阶段将相关历史纳入提示词以支持后续决策[21,22]。AgentBoard 等评测研究也将多轮交互轨迹作为分析智能体任务表现的重要依据[23]。也有研究尝试从步骤级价值评估角度改善长任务中的动作选择质量[24]。这一方式在短流程任务中较为有效，但在长任务场景下会迅速暴露局限性。由于最终目标往往需要经历大量相互依赖的中间步骤，智能体在执行过程中会积累冗长的工作记忆，若简单地将全部历史拼接进上下文，不仅会带来显著的令牌开销，也容易导致有效信息被冗余内容淹没，从而降低推理质量[25]。面向智能体生命周期学习的研究也指出，长期经验若缺乏筛选和组织，难以稳定转化为后续决策收益[26]。因此，如何提升长任务中的记忆利用效率，已经成为大语言模型智能体研究中的关键问题。

针对这一问题，近期研究开始尝试将智能体记忆进一步划分为工作记忆与长期经

验。前者主要保留当前任务阶段所需的局部上下文，后者则积累跨轮次、跨尝试形成的经验知识，并通过分层组织、摘要压缩和选择性检索来提高长任务中的信息利用效率[18,20]。面向适应性记忆和经验记忆的工作表明，智能体能力的提升不一定依赖持续更新模型参数，也可以通过改进文本形式的记忆组织与交互方式实现[27,28]。跨任务经验学习研究进一步说明，外部经验可以通过检索和复用参与后续任务决策[29]。然而，这类方法大多建立在集中式或资源较为充足的环境假设之上，对于资源受限边缘场景中的部署问题关注仍然不足。

与此同时，移动终端、边缘服务器与物联网设备上的智能服务需求持续增长，使得大语言模型智能体逐步从云端走向边缘[30,31]。边缘计算相关研究指出，靠近数据源和用户侧部署计算能力有助于降低传输时延、缓解带宽压力并增强本地服务能力[32]。端侧大模型适配和设备侧基础模型的发展，也进一步推动了大语言模型在边缘设备上的部署探索[33,34]。面向边缘智能的大模型综述指出，云平台上的大模型训练和推理面临成本、可扩展性和信息安全等挑战，端边侧协同训练与推理可降低时延和带宽需求[35]。在工业控制、具身交互、移动助理与实时协同等场景中，系统往往同时要求低时延、高响应性与较强的隐私保护能力，完全依赖云端推理会受到通信开销、链路波动和数据传输风险的制约[36,37]。工业互联网边缘智能研究同样强调，工业场景中的边缘智能需要围绕协同计算、资源隔离和隐私保护等关键技术展开[38]。因此，在资源受限的边缘环境中部署轻量化智能体已成为边缘智能的重要发展方向，围绕小参数语言模型与端侧基础模型的讨论也进一步推动了这一趋势[34,39]。可是在复杂长任务中，边缘节点又受制于算力、存储和上下文窗口，难以长期维持高质量推理与大容量历史维护[33,40]。这形成了边缘场景中的资源与能力矛盾，也构成了本文研究的现实背景。

1.2 研究目标

从现有研究进展来看，大语言模型智能体在长任务处理中的核心瓶颈可以概括为记忆表示与经验利用之间的矛盾。传统基于轨迹拼接的记忆方式虽然能够保留任务执行历史，但随着任务步数增长，工作记忆迅速膨胀，上下文中真正与当前决策相关的信息比例不断下降，进而导致令牌开销增大、检索效率降低以及决策质量波动[21,25]。换言之，问题并不只是历史信息不足，而是历史信息过多且难以有效组织。如果继续沿用统一的局部缓存来处理全部历史，就难以区分当前执行所需信息与可跨任务迁移的长期经验，从而削弱经验积累的价值[17,19]。

面向边缘部署时，上述矛盾会进一步受到资源条件和系统形态的放大。现有研究的不足主要体现在三个方面。首先，记忆机制多围绕单一节点或单一任务展开，能够缓解局部上下文压力，但缺少面向资源受限边缘长任务场景的系统性整合。其次，云边协同研究更多聚焦于计算卸载、资源调度和推理优化，能够回答模型推理应由谁承担，却较

少回答经验应由谁整理、如何沉淀以及何时反馈[30,40]。协同推理综述将边缘协同推理归纳为智能化方法与协同推理架构两类技术脉络，说明该方向主要围绕推理效率和动态场景下的资源协同展开[41]。协同训练研究则更多关注设备异构、资源受限和网络不稳定条件下的模型训练与参数更新机制[42]。最后，边缘侧在执行长任务过程中形成了大量有价值的轨迹数据，但这些数据若缺乏跨节点、跨任务共享的统一通路，就难以转化为可复用的外部经验；跨任务经验学习与智能体调优研究表明，经验的检索和复用是提升智能体泛化能力的重要方向[29,43]。开放环境下协作多智能体强化学习研究表明，多智能体协作需要面对环境要素变化、协作效率和泛化能力等问题[44]。由此形成的研究空白在于：现有工作尚未充分解决资源受限边缘环境中工作记忆控制、长期经验沉淀和反馈回流之间的协同问题。

基于上述分析，本文聚焦于资源受限边缘环境下大语言模型智能体的长任务处理问题，目标是在保持边缘侧低时延执行能力的同时，通过改进记忆组织方式与云边协同机制提升整体任务处理质量。具体而言，本文希望形成面向边缘长任务处理的云边协同方法思路，使边缘侧承担实时交互、子目标推进和工作记忆维护，云端侧承担阶段经验整理、长期经验沉淀和反馈增强；在此基础上，进一步设计边缘工作记忆、阶段摘要、云端长期经验处理和反馈回流机制，以降低长任务中的上下文冗余，提高跨轮次、跨任务经验的复用效率；最后，通过实验验证相关机制在任务成功率、执行效率与上下文消耗等方面的表现，分析其在边缘场景中的有效性与可部署性。

1.3 研究内容

针对边缘长任务处理中本地资源受限与经验利用不足并存的问题，本文采用云边协同的总体思路开展研究。受云边分层协同和交互记忆理论的启发[40,45]，本文将当前任务的在线决策与跨轮次经验的离线整理适度分离：边缘侧负责面向当前子目标进行实时感知、分步决策与本地执行，仅保留与当前阶段直接相关的边缘工作记忆；云端侧则接收边缘侧上传的结构化轨迹，对其中的成功经验、失败原因与策略模式进行集中分析与总结，并将整理后的经验信息在后续阶段反馈给边缘侧，以支持新的决策过程。通过这种职责划分，系统能够在资源受限条件下兼顾响应速度与经验增强能力。

围绕上述思路，本文的研究内容主要包括以下五个方面：

(1) 构建面向边缘长任务处理场景的云边协同方法架构。该架构以边缘侧在线执行和云端侧经验处理为两条主线：边缘侧直接与环境交互，围绕当前子目标进行动作生成、状态更新和阶段推进；云端侧不参与每一步实时动作生成，而是在边缘完成阶段任务后接收阶段摘要，进一步完成反馈生成、经验整理和跨节点共享。通过这一设计，系统将长任务中的即时决策与长期经验维护区分开来，使边缘节点能够在有限上下文和有限算力条件下持续推进任务。

(2) 设计面向长任务的分层记忆组织机制。围绕工作记忆、阶段摘要和长期经验三类信息，本文在边缘侧保留当前子目标相关的细粒度交互内容，用于支撑下一步动作生成；对于已经完成的阶段，则将原始轨迹压缩为阶段摘要，用于保留任务推进脉络并减少上下文冗余；对于跨阶段、跨节点仍具有复用价值的信息，则由云端进一步整理为长期经验。该部分研究重点在于回答“哪些信息应保留在边缘侧、哪些信息应上传云端、哪些经验应在后续阶段回流”这一核心问题。

(3) 设计云端评估、知识管理与反馈回流机制。云端接收边缘侧上传的阶段摘要后，一方面生成面向当前节点后续执行的单轨迹反馈，提示上一阶段中的有效策略、关键失误和后续调整方向；另一方面，当云端积累到来自多个节点或多个轮次的相似阶段经验时，进一步归纳可跨任务复用的全局指导。由此，边缘节点在后续阶段不仅能够利用本地工作记忆，还能够利用来自云端长期经验的补充信息，从而形成“边缘执行、阶段摘要、云端反馈、上下文更新”的闭环。

(4) 完成长任务处理方法原型与可视化支撑模块设计。方法原型用于实现边缘节点执行、阶段轨迹上传、云端反馈生成、长期经验维护和反馈回传等关键链路；可视化支撑模块则用于展示边缘节点状态、任务推进过程、阶段事件流和云端反馈状态，便于在实验过程中观察协同机制是否正常运行。该部分内容服务于方法实现和实验分析，使论文不仅停留在机制描述层面，也能够呈现实际运行过程。

(5) 开展实验验证与局限性分析。实验部分将选取具有明显多步依赖特征的代表性任务环境，围绕任务成功率、任务推进率、动作落地准确率、平均执行步数和上下文消耗等指标开展验证。通过对比实验考察云边协同机制对整体任务处理能力的影响，通过结果分析揭示边缘工作记忆与云端经验回流机制在长任务中的具体作用，并结合系统局限性分析讨论该研究在通信依赖、泛化能力和实时性权衡方面的边界条件。研究过程中，本文采用文献分析、系统设计与原型实现、实验验证相结合的方式，系统梳理大语言模型智能体、记忆机制、长任务处理与云边协同智能系统等领域的相关研究进展[4,17]。在此基础上，本文结合边缘智能体部署研究完成问题定位、机制设计和实验分析[31]。

1.4 论文结构

本文研究具有一定的理论意义和工程意义。理论上看，本文将大语言模型智能体的记忆管理问题进一步延伸到资源受限的边缘环境，强调通过边缘工作记忆控制、云端经验整理与反馈回流机制优化长任务处理过程，为边缘场景下智能体的持续决策与经验复用研究提供新的分析视角。工程上看，本文围绕边缘执行、云端经验管理、反馈传递和可视化监测构建系统原型，关注低时延、强交互和资源受限条件下的实际部署需求，对

于构建可落地的边缘智能体系统、提升其在长任务中的稳定性与处理效率具有一定参考价值。

全文共分为 5 章，各章内容安排如下：

- 第 1 章为引言，介绍研究背景、问题来源、研究目标、研究内容、研究意义以及全文结构安排。
- 第 2 章为相关研究与理论基础，梳理国内外研究现状，综述大语言模型智能体、智能体记忆机制、云边协同智能系统等相关研究，并结合认知记忆理论和云边分层职责明确本文的研究定位。
- 第 3 章为云边协同长任务处理方法设计与实现，详细介绍方法总体架构、边缘侧执行机制与本地记忆设计、云端评估机制与知识管理设计、云边协同反馈机制、跨边缘经验共享，以及可视化支撑模块的设计与实现。
- 第 4 章为实验设计与结果分析，给出实验场景、任务设置、评价指标、实现细节与实验参数设置，并通过两个实验场景和消融实验分析云端反馈与强节点经验对弱模型边缘节点的影响。
- 第 5 章为结论与展望，总结本文研究工作，讨论实际意义，并对后续研究方向进行展望。

2 相关研究与理论基础

2.1 国内外研究现状

围绕资源受限边缘环境下的大语言模型智能体长任务处理问题，已有研究大致可以从大语言模型智能体、智能体记忆机制和云边协同智能系统三个方向进行梳理。国外研究较早围绕通用智能体框架、工具调用、环境交互和长任务评测展开，重点关注模型如何在开放环境中完成推理、行动和反馈闭环。例如，ReAct 将推理过程与动作执行交替组织，为后续智能体任务执行提供了基础范式[21]。Voyager 等开放环境智能体进一步展示了大语言模型在持续探索、技能积累和任务推进中的潜力[13]。围绕长任务执行和记忆管理，HiAgent 等工作开始关注分层工作记忆与上下文压缩对长时程任务的作用[25]。Memento 进一步尝试在不微调底层大语言模型的条件下，通过经验记忆改进智能体能力，说明外部经验组织也可以成为智能体适应的重要路径[28]。

国内研究则更多从综述和系统化梳理角度总结相关方向的发展脉络。大型语言模型综述从模型原理、实现路径和发展趋势出发，总结了上下文学习、指令遵循和复杂推理等能力基础[3]。基于大语言模型的智能体综述进一步将智能体的关键组成概括为感知、规划、记忆和行动等模块，并讨论了其应用场景与未来问题[6]。在边缘智能方向，边缘计算综述强调了近数据源处理、低时延和带宽节约等基础优势[32]。面向边缘智能的大模型研究进一步指出，端边云协同训练与推理是缓解云端大模型成本、时延和安全压力的重要方向[35]。协同推理综述从推理任务划分、模型切分和任务卸载等角度总结了云边协同推理路径[41]。协同训练综述则从边缘侧模型训练、资源受限和设备异构等角度总结了云边训练分工[42]。

总体来看，国内外研究已经分别在智能体执行框架、记忆管理方法和云边协同计算方面形成了较丰富成果。已有发现说明，大语言模型智能体需要在感知、规划、行动和反馈之间形成闭环；长任务执行需要控制上下文膨胀并对历史经验进行筛选；边缘智能系统则需要在低时延、资源受限和隐私保护之间进行权衡。然而，现有研究仍较少把三者结合起来讨论：智能体研究较少面向边缘资源约束，记忆机制研究较少考虑跨节点经验沉淀，云边协同研究又更多关注计算协同和推理优化。本文正是在这一研究空白下，进一步关注边缘侧工作记忆控制、云端长期经验组织和反馈回流之间的协同问题。

2.2 大语言模型智能体相关研究

2.2.1 基本概念与运行形式

大语言模型智能体是指以大语言模型为核心推理单元，能够结合环境观测、历史上下文与任务目标持续生成动作并完成复杂任务的智能系统[4,5]。中文智能体综述也将这类系统概括为由感知、规划、记忆和行动等模块共同构成的智能体形态[6]。与传统静态文本生成模型相比，这类系统强调感知、决策、执行与反馈之间的闭环过程，因此在问题求解形式上更加接近真实世界中的连续决策任务。近年来，随着大语言模型推理能力的提升，相关研究逐步由单轮对话与问答扩展到工具调用和开放环境探索等任务形态[8,13]。WebGPT 等工作说明，浏览器辅助的信息检索和人类反馈能够扩展语言模型在开放问答任务中的能力边界[14]。国内大语言模型综述也从模型原理、实现路径和发展趋势角度说明了上下文学习、指令遵循和复杂推理能力对智能体系统形成的基础作用[3]。在具体应用方面，相关工作已经覆盖软件工程和机器人系统等场景[9,10]。大语言模型用户模拟框架进一步表明，智能体方法可以服务于对话推荐等交互式系统评测[11]。具身智能综述表明，大模型驱动的智能系统正在向感知环境、规划任务和执行动作的闭环形态发展[15]。基于大模型的具身智能系统综述进一步从系统组成、感知决策和执行反馈角度总结了该方向的发展[16]。这些工作表明，大语言模型智能体已经不再局限于生成答案，而是开始承担任务分解、策略规划和自主执行等更高层次的功能。

从基本运行形式看，大语言模型智能体通常可以被抽象为一个循环决策过程。给定任务目标、当前环境观测和历史上下文，智能体在每一轮生成动作，并根据环境反馈更新后续决策所需的信息。若记时刻 t 的环境观测为 o_t，任务目标为 g，历史上下文为 h_t，则智能体动作生成过程可概括为

$$a_t = \pi_\theta(o_t, g, h_t), \tag{2-1}$$

式中 a_t ——智能体在时刻 t 输出的动作；

o_t ——时刻 t 的环境观测；

g ——智能体需要完成的任务目标；

h_t ——时刻 t 之前积累的历史上下文；

$\pi_\theta(\cdot)$ ——由大语言模型及其提示、工具和记忆组件共同构成的决策过程。

该表达式并不限定具体模型结构，而是用于说明智能体与普通文本生成系统的关键差别：智能体的输出并非孤立文本，而是会作用于环境并影响下一轮状态的动作。第 3 章中边缘侧动作生成公式正是在这一抽象基础上进一步加入子目标与边缘本地记忆。

2.2.2 方法演进与系统组成

从方法脉络来看，当前大语言模型智能体研究主要由开放基准、通用智能体框架和工具增强方法推动[4,5]。AgentGym 等评测与训练基准进一步推动了跨环境智能体能力评估与训练研究[46]。中文智能体综述进一步从体系结构、能力评测和应用场景角度对该方向进行了系统梳理[6]。总体而言，已有研究已经从单一模型能力评测逐步转向面向任务过程的系统能力分析，即不只关注模型能否给出正确答案，也关注智能体能否在多轮交互中持续感知环境、更新状态并修正动作。

在长任务评测方面，AgentBoard 提供了面向大语言模型智能体的综合评测框架，将规划、文本交互等任务组织为可执行环境，并通过成功率、任务推进率和动作落地准确率等指标刻画智能体的过程表现[23]。HiAgent 则在 AgentBoard 评测环境与任务口径基础上进一步组织长时程任务执行流程，并围绕分层记忆与上下文压缩展开方法设计[25]。这一类工作为比较不同智能体执行策略提供了统一任务入口和指标口径，也为本文后续实验选择任务环境、边缘执行框架与评价指标提供了参考。

从方法演进来看，现有大语言模型智能体研究大致沿着两个方向展开。一类工作聚焦于提升单个智能体的规划、推理和环境适应能力。例如，ReAct 将推理过程与动作生成交替组织，使模型能够在思考与行动之间建立更加紧密的联系，从而改善复杂任务中的决策质量[21]。随后，一些研究进一步引入环境生成和任务生成机制，以增强智能体在开放环境中的适应性与泛化能力[7]。社会网络仿真系统则展示了大语言模型智能体在开放式社会交互建模中的应用潜力[47]。另一类工作则关注多智能体协作，通过角色分工、过程通信和任务编排提升系统在复杂任务中的整体表现[48,49]。AgentNet 进一步从去中心化演化协调角度讨论大语言模型多智能体系统中的协同优化问题[50]。开放环境下协作多智能体强化学习综述也表明，开放环境中的多主体协作需要同时面对环境要素动态变化、协作效率和策略泛化等问题[44]。这类研究强调多个智能体之间的信息流动与责任划分，为解决单一智能体能力受限的问题提供了重要思路。

从系统组成看，大语言模型智能体通常包括任务理解、规划决策、记忆管理、工具调用和执行反馈等模块。任务理解模块负责将用户指令或环境目标转化为可操作的任务表示；规划决策模块负责将目标拆分为若干可执行步骤；记忆管理模块负责保存当前任务状态、历史轨迹和可复用经验；工具调用模块负责将语言模型的决策转化为搜索、计算、代码执行或环境控制等外部操作；执行反馈模块则将环境返回的信息重新纳入后续决策。由此可见，智能体能力并不只来自大语言模型本身，也来自围绕模型建立的外部状态管理和动作执行机制。这一事实为本文关注记忆组织和云边协同提供了依据：当智能体进入边缘长任务场景后，系统瓶颈往往不再只是模型能否理解指令，还包括历史状态如何维护、阶段经验如何沉淀以及外部反馈如何回到决策过程。

2.2.3 边缘长任务场景中的局限

尽管相关研究显著拓展了大语言模型智能体的应用边界，但多数方法仍默认较为充足的算力、存储和上下文资源。对于需要在边缘环境中持续处理长任务的系统而言，模型推理能力、上下文维护能力和资源消耗之间存在更为尖锐的矛盾。尤其是在边缘设备或边缘服务器上，端侧大模型部署会受到模型规模和资源预算限制[31,39]；当任务进一步延伸为多轮长时序交互时，系统还需要在有限上下文内维护阶段状态并持续调整策略。面向边缘智能的大模型研究指出，云平台的大模型训练和推理存在成本、时延、可扩展性和信息安全等挑战，端边云协同是缓解这些问题的重要方向[35]。现有智能体研究的主要缺口在于，它较多讨论如何提升智能体本身的规划与执行能力，却较少讨论当智能体进入边缘长任务场景后，系统应如何在有限上下文、有限存储和持续交互之间分配状态维护责任。因此，引言部分提出的边缘长任务处理问题，实质上是在大语言模型智能体研究进一步走向真实部署场景后所暴露出的关键瓶颈。这一问题也决定了本文后续研究不能停留在通用智能体框架层面，而需要进一步结合记忆管理和云边协同机制展开系统设计。

2.3 智能体记忆机制与长时序任务研究

2.3.1 记忆机制的功能划分

记忆机制是大语言模型智能体区别于普通大语言模型的重要特征之一，它决定了系统如何保留历史信息、理解任务上下文并在后续步骤中利用既有经验支持决策[5,17]。从认知功能角度看，智能体在执行任务时至少需要同时处理两类信息：一类是与当前任务阶段直接相关的工作记忆，另一类是跨轮次、跨任务积累形成的长期经验。前者通常决定当前动作是否合理，后者则影响系统能否在重复试错中持续改进[18,20]。围绕长期对话、生命周期记忆和智能体能力演化等问题，已有研究进一步讨论了记忆内容的时间组织与经验复用方式[19,51]。因此，如何在有限上下文中保留关键状态、如何将过去经验转化为未来决策依据，已成为长任务智能体领域的核心议题。

从形式上看，智能体记忆可以被理解为当前决策可访问的信息集合。若将时刻 t 的记忆表示为 M_t，则其可进一步拆分为服务即时决策的工作记忆 M_t^{work} 与服务经验复用的长期经验 M^{long}：

$$M_t = \left\{ M_t^{\text{work}}, M^{\text{long}} \right\}. \tag{2-2}$$

式中 M_t ——时刻 t 智能体当前可访问的记忆集合；

M_t^{work} ——服务即时决策的工作记忆；

M^{long} ——服务经验复用的长期经验。

其中，M_t^{work} 随当前任务阶段变化，通常包含最近观测、当前子目标、短期动作历

史和局部状态；M^{long} 则跨轮次或跨任务保持相对稳定，通常包含历史成功策略、失败原因、阶段摘要和可检索案例。式 (2-2) 的意义在于明确不同记忆类型的作用边界：工作记忆解决当前步骤 ‘‘如何继续执行’’ 的问题，长期经验解决后续阶段 ‘‘已有经验如何复用’’ 的问题。第 3 章将进一步把这一划分落到边缘本地记忆和云端长期经验的系统结构中。

2.3.2 长任务记忆管理方法

围绕这一议题，国内外研究主要形成了三类代表性思路。第一类是基于轨迹的短期记忆管理，即将观测与动作序列作为工作记忆的主要内容，在推理时直接拼接到上下文中，以维持任务连续性[22,23]。进一步的动作价值建模工作尝试在步骤级别评估不同决策的潜在收益，从而改善长任务中的动作选择质量[24]。这类方法实现简单，能够较好适配短流程任务，但随着任务步数增加，轨迹长度持续膨胀，容易出现上下文冗余、信息淹没与推理效率下降等问题。

第二类是分层记忆与摘要压缩方法，其基本思想是在保留当前阶段细粒度信息的同时，将已完成阶段的交互内容进行抽象和压缩，从而降低长任务中的上下文压力[25,27]。这类工作表明，将长任务拆分为若干局部阶段，并对非当前阶段信息进行摘要，是提升长任务成功率的重要手段。也有研究指出，长任务中的上下文管理收益并不一定完全来自复杂摘要，更轻量的观测屏蔽策略在某些场景下也可能取得接近效果，这说明记忆组织的关键在于信息选择而非单纯压缩[52]。第三类是跨任务经验检索与案例复用方法，即将历史成功或失败轨迹转化为可复用经验，在后续任务中通过检索相似经验辅助决策[26,29]。Memento 也表明，智能体可以在不微调底层大语言模型的情况下，通过记忆化经验和反馈进行能力适配[28]。这类方法使智能体能够在不更新模型参数的情况下，通过外部记忆持续积累能力。

2.3.3 阶段化轨迹与边缘部署问题

长任务场景进一步放大了记忆管理的重要性。若一项任务由 K 个阶段组成，每个阶段形成一段局部轨迹，则完整历史可表示为若干阶段轨迹的集合：

$$\tau = \{\hat{\tau}_1, \hat{\tau}_2, \ldots, \hat{\tau}_K\}. \tag{2-3}$$

式中 τ ——一项长任务执行过程中形成的完整阶段轨迹集合；

$\hat{\tau}_j$ ——第 j 个任务阶段形成的局部轨迹；

K ——长任务被划分后的阶段数量。

当系统直接把 τ 中所有历史内容拼接进入提示词时，输入长度会随 K 增大而持续增长，早期阶段中的无关细节也会挤占当前决策所需的上下文空间。因此，长任务记忆

组织的核心不应是无限扩展历史缓存，而应是围绕阶段边界对 $\hat{\tau}_j$ 进行筛选、压缩和归档。第 3 章中的阶段摘要正是基于这一思想，将已完成阶段的局部轨迹映射为更紧凑的摘要表示。

总体来看，现有记忆研究已经从历史全量拼接发展到分层组织、压缩和检索，为长任务处理提供了更加细粒度的设计空间。然而，这些研究大多围绕单机环境或资源相对充足的集中式场景展开，对资源受限边缘部署下的记忆组织问题关注不足。更具体地说，现有工作虽然能够说明历史应被压缩或检索，却没有充分回答三类问题：边缘侧应保留多少工作记忆，已完成阶段应以何种粒度转化为阶段摘要，跨节点积累的长期经验又应如何回流到后续决策。一方面，边缘节点难以像中心化系统那样长期维护大规模、多轮次、细粒度的历史信息；另一方面，纯本地执行模式使各边缘节点只能依赖自身经验，难以共享跨节点、跨任务的有效知识。由此可见，后续系统设计需要重点解决三个问题：如何控制边缘侧工作记忆规模、如何将阶段性轨迹转化为可复用摘要、以及如何让长期经验在云边之间形成稳定回流。

2.4 云边协同智能系统相关研究

2.4.1 云边协同的基本问题

云边协同智能系统的核心问题在于如何在端、边、云等不同层次之间分配计算任务与智能能力，以兼顾实时性、计算效率和服务质量[30,36]。边缘计算综述指出，边缘侧靠近数据源和用户，适合承担低时延、带宽敏感和隐私敏感的服务任务[32]。在大语言模型应用场景中，云端通常具备更强的模型推理能力和更大的存储空间，而边缘侧更接近用户与环境，能够提供更低时延的交互响应。基于这一特点，现有研究普遍认为，完全依赖云端推理虽然能够获得更强的推理能力，但会带来较大的通信时延和数据传输成本；完全依赖边缘执行虽然能够保障响应速度，却会受到模型规模和资源预算的限制[31,33]。同时，传统云端与设备端协同研究也从模型传输、服务部署和资源分配等角度讨论了边缘智能系统的性能约束[53]。工业互联网边缘智能研究进一步说明，工业场景下的边缘智能系统需要关注协同计算、资源隔离、隐私保护和智能运维等关键技术[38]。因此，云边协同成为提升边缘大语言模型系统可部署性的关键路径。

从事实依据看，大模型向边缘侧部署面临的压力主要来自三个方面。第一，模型规模扩大使推理计算和显存需求持续上升，边缘节点难以独立承担完整模型的高频推理；第二，长任务智能体需要维护多轮交互历史和阶段状态，进一步增加上下文和存储压力；第三，真实边缘场景通常存在网络波动、设备异构和隐私约束，系统不能简单假设所有数据都可以稳定上传到中心云端。因此，边缘侧与云端之间需要形成分层职责：边缘侧承担低时延交互和局部决策，云端侧承担更重的经验整理、跨节点汇聚和长期存

储。这一事实基础决定了本文中的云边协同不是单纯的模型卸载，而是围绕长任务经验流动建立的系统分工。

2.4.2 资源调度与协同推理研究

从国内外研究现状来看，云边协同领域已有较充分的计算协同和资源调度研究，研究重点主要集中在服务放置、推理加速、模型压缩和边缘资源管理等方面。协同推理综述将边缘智能中的推理协同归纳为智能化方法和协同推理架构两类技术路线[41]。随着生成式人工智能和大语言模型进入移动网络与边缘设备，相关研究开始进一步关注大模型推理任务在云端、边缘侧和端侧之间的划分方式[30,31]。不过，这类研究更多围绕推理效率和服务质量展开，对长任务智能体执行过程中产生的阶段经验如何整理、沉淀和回流仍讨论不足。

在已有工作中，一类研究主要从资源调度和推理优化角度展开，关注如何通过任务卸载、服务部署和推理复用来提高整体性能。例如，针对移动边缘环境的研究指出，合理的服务部署与资源分配能够有效提升边缘智能系统的响应效率[36]。面向大语言模型推理的研究则尝试通过跨任务复用中间状态、压缩计算开销等方式提升边缘推理能力[37,39]。Division-of-Thoughts（DoT）进一步利用大小模型协同，将混合语言模型能力用于端侧智能体，从而在端侧效率与任务能力之间取得折中[54]。另一类研究进一步将云边协同拓展到大语言模型任务处理过程之中，强调云端与边缘侧在任务执行中承担不同职责，例如边缘侧负责即时响应，云端侧负责更复杂的推理与优化[30,40]。与协同推理相对应，协同训练研究则关注边缘智能场景下训练数据、模型更新和计算资源在多层节点之间的协同分配，并强调设备异构、资源受限和网络不稳定对训练过程的影响[42]。这些工作从系统层面说明，边缘智能体并不一定需要独立完成全部计算过程，而可以借助云端能力弥补资源短板。

2.4.3 从计算协同到知识协同

若从协同对象来划分，现有云边协同大致可分为计算协同、数据协同和知识协同三类。计算协同关注模型推理或训练任务在云端、边缘和终端之间的分配；数据协同关注数据采集、传输、缓存和隐私保护；知识协同则进一步关注边缘节点产生的经验如何被汇聚、组织并重新服务后续任务。本文关注的长任务处理问题更接近第三类：边缘节点本地执行会产生大量阶段轨迹，这些轨迹如果只保留在本地，难以形成跨节点复用；如果不加筛选地上传云端，又会带来额外通信和存储压力。因此，云边协同需要在阶段摘要、经验归档和反馈回流之间建立稳定机制。

不过，现有云边协同研究整体上仍更偏向计算协同而非认知协同。多数工作着眼于如何减少延迟、降低带宽开销或优化推理路径，而对如何利用云端能力帮助边缘智能体

开展经验总结、错误修正和知识沉淀讨论较少[31,40]。VELO 等工作已经开始引入向量数据库辅助云边协同优化大语言模型服务质量，说明向量化存储可以用于云边系统中的信息组织与检索[55]。但这类研究主要面向服务质量优化，而非长任务执行过程中经验的总结、复用与跨节点传播。换言之，现有系统往往把云端视为额外算力来源，却较少将其进一步视为长期经验处理与共享知识形成的中心节点。这一缺陷使云边协同容易停留在推理链路优化层面，难以支撑长任务中持续产生的阶段经验被整理、复用和跨节点传播。对于引言部分提出的长任务处理问题而言，仅解决由谁计算还不够，更关键的是解决经验如何沉淀、反馈如何回流以及跨节点知识如何共享。因此，后续系统设计不仅要处理任务执行链路，还要建立云端评估、知识组织和跨节点共享之间的完整闭环。

2.5 理论基础与本文研究定位

结合以上相关研究可以看出，大语言模型智能体、记忆机制和云边协同智能系统三条研究主线分别回答了智能体如何执行任务、历史经验如何支持长任务决策、云端与边缘如何协同分工三个关键问题，但三者在资源受限边缘长任务场景中的结合仍不充分。智能体研究为长任务中的规划与执行提供了基本框架，记忆研究为工作记忆控制与长期经验复用提供了方法基础，云边协同研究则为系统部署和能力分层提供了工程路径。然而，三类研究之间仍存在明显断点：智能体研究较少面向边缘资源约束，记忆研究较少考虑云边分层部署，云边协同研究又较少触及经验总结与认知反馈。由此产生的研究空白，是缺少一个能够同时面向边缘实时执行、长任务经验积累以及跨节点知识回馈的统一系统视角。

2.5.1 认知记忆理论基础

本文的理论基础首先来源于认知科学中关于工作记忆与长期记忆的区分。将这一划分迁移到大语言模型智能体研究中，可以看到已有综述和长期对话记忆研究也常从短期上下文维护、长期经验保存和后续检索利用等角度讨论智能体记忆结构[19,51]。这一认识为资源受限边缘场景下的记忆组织提供了直接启发：边缘侧更适合保留与当前子目标密切相关的工作记忆，而不必持续维护完整长轨迹；跨阶段、跨任务仍有价值的信息，则更适合以压缩后的长期经验形式沉淀下来，作为后续决策时可再次利用的外部支持。由此，工作记忆与长期经验的区分不仅是概念层面的划分，也构成了后续系统进行分层记忆组织的基本依据。

进一步说，工作记忆与长期经验之间并不是简单的存储容量差异，而是信息功能的差异。工作记忆强调即时性、局部性和任务相关性，适合服务当前动作生成；长期经验强调稳定性、可迁移性和可检索性，适合服务后续任务中的策略选择。对于边缘智能体而言，这一区分具有直接工程含义：边缘侧不应承担全部历史的长期维护，否则会受到

上下文长度和存储资源限制；云端也不应接管每一步实时动作生成，否则会削弱边缘侧低时延响应优势。因此，认知记忆理论为‘‘边缘保留工作记忆、云端沉淀长期经验’’的系统分工提供了理论依据。

2.5.2 长任务中的记忆组织原理

记忆驱动的智能体研究进一步表明，历史经验并非只能以原始轨迹形式存在，而可以通过摘要压缩、结构化组织和检索增强等方式转化为可复用的信息资源，从而提升后续决策效率[25,27]。跨任务经验学习研究进一步说明，历史经验在经过结构化整理后可以支持后续任务中的案例复用与策略迁移[29]。对于长任务而言，这一点尤为重要。相关研究指出，大语言模型智能体在复杂环境中往往不是一次性完成全局目标，而是围绕若干子目标逐步推进，并在每一阶段根据环境反馈修正后续动作[21,25]。这意味着，长任务中的有效信息并不平均分布在全部历史轨迹中，而更多集中于与当前阶段直接相关的工作记忆，以及能够影响后续阶段选择的关键经验。

进一步来看，历史信息的价值不仅取决于是否被保存，还取决于其是否能够以适合调用的形式进入后续决策过程。已有研究显示，原始轨迹虽然信息完整，但往往包含大量与当前推理无关的细节；相比之下，经过阶段摘要、结构化表示或相似性检索处理后的经验，更容易在后续任务中被快速调用[25,27]。面向智能体持续学习的研究也强调，长期经验需要经过筛选、归纳和检索机制才能在后续任务中发挥作用[26,29]。因此，从理论上说，长任务中的记忆利用可以被理解为信息筛选与重组过程：系统不是被动积累全部历史，而是主动保留对未来决策最有价值的部分。对于资源受限边缘场景而言，这一点尤为重要，因为上下文预算、存储容量和推理时延都要求记忆系统具备较强的压缩与选择能力。换言之，长任务中的记忆组织重点不在于保存完整历史，而在于围绕当前子目标保留必要工作记忆，并将已完成阶段转化为可复用的阶段摘要和长期经验。

基于上述分析，长任务中的上下文组装可以被抽象为对当前观测、子目标、工作记忆和外部经验的选择性组合。若记时刻 t 的决策上下文为 P_t，当前子目标为 g_t^{sub}，外部经验反馈为 E_t，则有

$$P_t = \mathscr{A}\left(o_t, g_t^{\mathrm{sub}}, M_t^{\mathrm{work}}, E_t\right), \tag{2-4}$$

式中 P_t ——时刻 t 输入智能体的决策上下文；

o_t ——时刻 t 的环境观测；

g_t^{sub} ——时刻 t 当前正在执行的子目标；

M_t^{work} ——时刻 t 可用的工作记忆；

E_t ——时刻 t 可用的外部经验反馈；

$\mathscr{A}(\cdot)$ ——上下文组装过程。

该公式强调，进入模型推理的不应是未经筛选的完整历史，而是由当前观测、当前

子目标、必要工作记忆和可用外部经验共同构成的压缩上下文。第 3 章中云端反馈注入公式将进一步把 E_t 具体化为云端回流的单轨迹反馈和全局指导。

2.5.3 云边协同中的分层职责与反馈闭环

云边协同研究说明，在分层计算环境中进行职责划分，有助于在低时延执行与较强推理能力之间取得平衡[30,36]。从系统运行角度看，这种协同方式不仅涉及算力分配，也可以被扩展为面向经验处理的异步反馈闭环。边缘侧负责与环境持续交互，需要优先保证当前阶段的响应连续性；云端侧具备更集中的计算和存储条件，因而适合承担更重的分析、归档与经验整理任务。相关云边大语言模型服务研究已经说明，云端与边缘侧可以围绕服务质量和推理职责进行分工[40]；本文进一步将这种分工用于长任务阶段经验的整理与回流。

结合交互记忆理论可以进一步看出，云端在这里并不是边缘执行过程的简单外部算力补充，而更接近一种外部认知支持结构[45]。从智能体长期记忆研究的角度看，经验需要以更稳定、更可检索的形式保存，才能在后续任务中继续发挥作用[19,51]。据此，本文将边缘侧定位为当前任务工作记忆的维护者，将云端定位为长期经验的组织者，并在适当时机将其以反馈形式送回边缘侧。因此，云边协同在本文所讨论的长任务场景中，不仅意味着计算资源如何放置，也意味着阶段摘要由谁整理、长期经验如何形成、反馈信息何时回流以及跨节点知识如何共享。

由此可以将本文所需的反馈闭环概括为四个环节：边缘侧围绕当前子目标进行在线执行；子目标完成后，边缘侧将局部轨迹压缩为阶段摘要；云端对阶段摘要进行评估和归档，形成单轨迹反馈或全局指导；反馈信息再回到边缘侧，参与后续阶段的上下文组装。该闭环的关键不在于让云端替代边缘实时决策，而在于让已完成阶段的经验能够及时转化为后续阶段可用的信息。与单纯任务卸载相比，这种协同方式更强调经验流动和知识积累，也更符合长任务智能体持续执行和持续改进的需求。

基于上述理论基础，本文不再单纯从模型规模或任务卸载角度优化系统，而是聚焦资源受限边缘场景中的长任务处理问题，围绕边缘工作记忆控制、阶段轨迹经验化和云端反馈回流构建云边协同处理方法。

2.6 本章小结

本章围绕大语言模型智能体、智能体记忆机制与云边协同智能系统三条主线展开综述，并从认知记忆划分、长任务记忆组织和云边反馈闭环三个方面明确本文的理论基础。现有研究在智能体执行、记忆利用和云边协同方面提供了重要启发，但对边缘资源约束、跨节点经验沉淀与经验回流机制关注仍不足。因此，后续将围绕边缘工作记忆控制、云端长期经验组织和反馈回流展开方法设计。

3 云边协同长任务处理方法设计与实现

3.1 总体架构设计

3.1.1 设计目标

结合前文对长任务处理问题、智能体记忆机制以及云边协同思路的分析，本文进一步给出面向边缘场景的协同设计。前文已从智能体动作生成、记忆功能划分、阶段化轨迹和上下文组装等角度给出了基本抽象；本章在此基础上进一步说明这些抽象如何落到具体的云边协同架构中。对于由多个相互依赖阶段组成的长时程任务，若边缘节点长期维护完整历史，不仅会增加上下文负担，也会削弱当前阶段相关信息在决策中的作用。

因此，本章的设计目标是将长任务中的在线决策与经验处理适度解耦。已有边缘大模型研究表明，边缘侧部署需要同时考虑低时延执行和资源约束[30,31]；云边协同研究进一步说明，云端与边缘侧可以围绕服务质量和推理职责形成分层协作[40]。在此基础上，本文进一步将这种分工用于长任务经验处理：边缘侧负责围绕当前子目标持续执行，并维护支撑即时决策的工作记忆；云端侧负责对已完成阶段的轨迹进行总结、整理与回流，并逐步形成可复用的长期经验。通过这种分工，边缘节点不需要在每一步都依赖云端动作生成，而是将阶段性轨迹作为经验组织、云端处理和反馈回流的基本单位。

基于上述目标，后续内容先从整体工作流程说明边缘侧与云端侧如何配合，再进一步展开总体结构中的关键模块和数据流向。

3.1.2 工作流程

本文方法的整体流程如图 3-1 所示。该图从宏观层面展示了用户请求、边缘节点集群和云端节点之间的关系：左侧用户发起任务请求，中部边缘节点集群承担实际交互与执行，右侧云端节点负责轨迹分析、经验整理和知识回流。该图的作用是先给出协同过程的整体轮廓，为后续总体结构中的详细数据流说明提供入口。

图 3-1 表明边缘侧并不是单个孤立智能体，而是由多个大语言模型智能体构成的节点集群。每个边缘智能体内部包含规划器（Planner）、执行器（Executor）以及本地记忆，用于围绕当前任务与交互环境完成观察、决策和动作执行。当某一阶段完成后，边缘节点会将该阶段形成的轨迹上传至云端，而不是在每一步动作后都请求云端参与。

云端侧接收来自不同边缘节点的阶段轨迹后，通过大语言模型评估器对轨迹进行分析，形成摘要、经验提示和评估结果，并将可复用内容写入知识向量库。随后，云端将整理后的结果回流至边缘节点，用于后续阶段的上下文构造。图中从云端返回边缘侧的箭头表示这种经验注入关系：云端反馈并不替代边缘侧的实时决策，而是作为外部经验

补充进入后续执行过程。

因此，本文方法的整体流程可以概括为边缘执行、阶段摘要、云端反馈和上下文更新四个步骤。首先，边缘节点围绕当前子目标执行观察、决策、动作和反馈的交互循环，并在本地维护支撑即时决策的工作记忆。其次，当某一子目标结束后，边缘侧将该阶段轨迹压缩为结构化的阶段摘要，并异步上传至云端。然后，云端针对该阶段轨迹生成单轨迹反馈，并结合已有长期经验形成更高层次的全局指导。最后，边缘节点在后续决策轮次中将这些云端结果作为附加上下文注入本地推理过程。

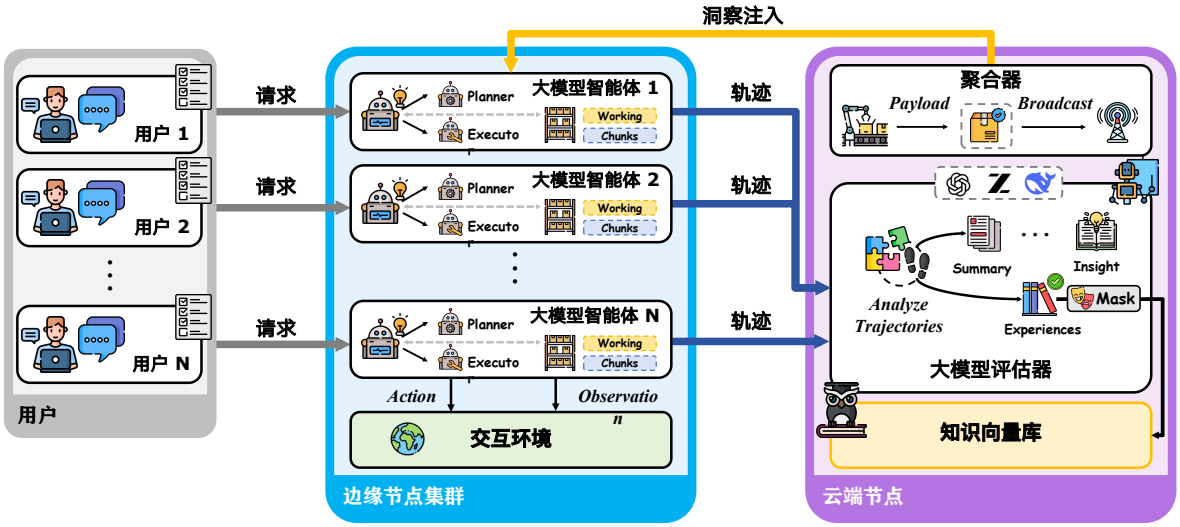


图 3-1　云边协同方法工作流程概览

前文中式 (2-3) 将长任务表示为若干阶段轨迹的集合。与这一表示相对应，图 3-1 中边缘节点上传到云端的并不是完整历史，而是围绕已完成子目标形成的局部轨迹或阶段摘要。也就是说，本文在结构设计上沿用阶段化轨迹的处理思想，以局部阶段作为摘要、上传和反馈回流的基本单元。下一小节进一步说明边缘侧和云端侧内部各模块如何承载这些数据流。

3.1.3　总体架构

图 3-2 给出了本文方法的详细架构。相比图 3-1 的整体流程概览，该图进一步展开了边缘侧和云端侧内部的具体组成，并在左侧图例中标出了五类通道：轨迹上传、总结请求与响应、全局指导注入、记忆注入以及确认回执。为便于理解，下面按照总体划分、边缘侧模块、云端侧模块和通道数据流四个层次对该图进行说明。

首先，从总体划分看，图 3-2 将架构分为边缘侧和云端侧两部分。边缘侧以大语言模型智能体为执行主体，直接接收用户任务并与交互环境相连，负责完成面向当前子目

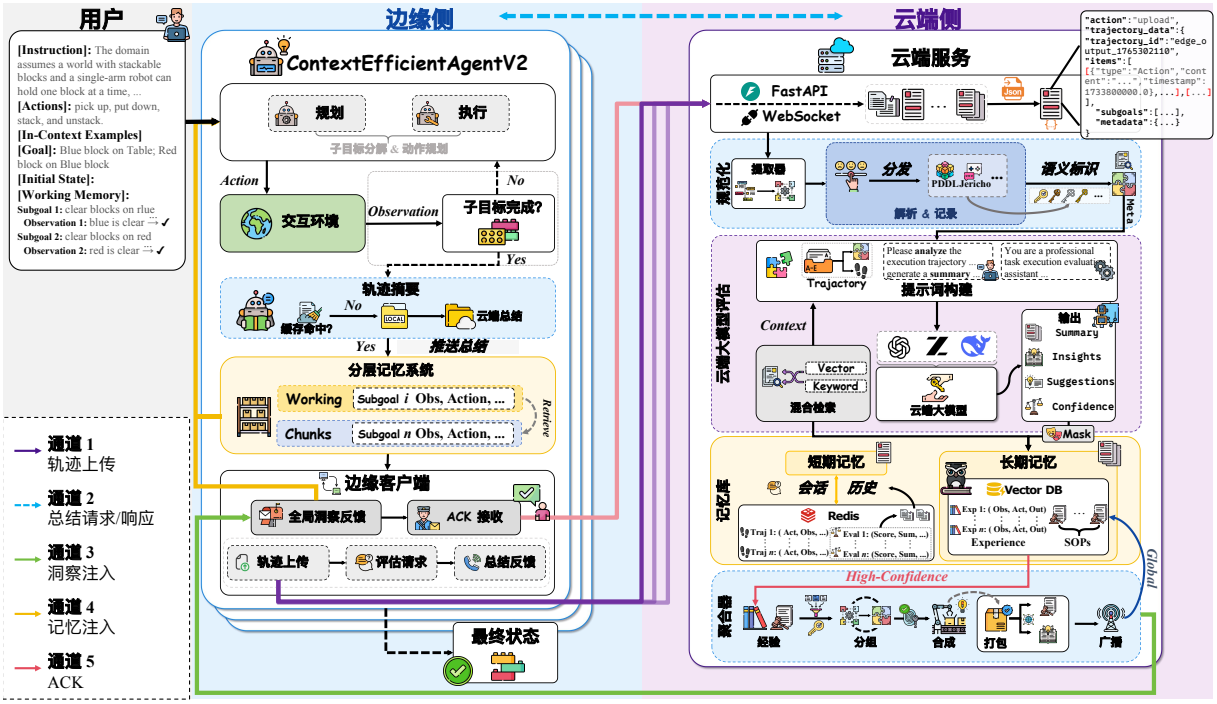


图 3-2　本文方法总体架构图

标的在线执行。云端侧位于图的右侧和下方，主要包括云端服务、云端大模型评估、记忆库和聚合器等部分，用于接收边缘上传的轨迹，生成阶段性总结与经验反馈，并将可复用经验沉淀为后续可调用的长期知识。

在边缘侧，图中核心执行单元标为 ContextEfficientAgentV2，其内部包含规划和执行两个子模块。规划模块根据用户指令、上下文示例和当前工作记忆形成子目标分解与动作规划；执行模块将规划结果转化为环境动作，并根据环境返回的观测判断当前子目标是否完成。若子目标尚未完成，边缘侧继续在本地循环执行；若子目标已经完成，则进入轨迹摘要阶段。轨迹摘要部分先检查本地缓存是否命中，若命中则直接使用本地已有总结，若未命中则向云端发起总结请求。随后，阶段摘要被写入分层记忆系统，其中 Working 部分保存当前子目标的观测和动作记录，Chunks 部分保存已完成子目标的压缩片段。

在云端侧，云端服务通过 FastAPI 和 WebSocket 接收边缘侧上传的数据，并将原始请求转换为结构化记录。图中规范化部分包含提取器、分发和语义标识等步骤，用于解析轨迹内容、区分 PDDL 与 Jericho 等任务类型，并生成后续评估所需的任务、阶段和关键信息。云端大模型评估部分基于轨迹、提示词构建和混合检索结果生成输出，输出内容包括 Summary、Insights、Suggestions 和 Confidence。

云端记忆库由短期记忆和长期记忆组成。短期记忆依托 Redis 保存会话历史、轨迹记录和评估结果，主要服务近期任务中的快速回流；长期记忆依托向量数据库（Vector DB）保存经验条目和全局指导。系统实现和界面中仍保留 SOPs、Insights 等字段名，用

于标识候选全局指导或其展示载荷；论文叙述中统一称为全局指导。聚合器位于云端侧右侧，按照选择、分组、合成和广播四个步骤处理高置信度经验：先从长期经验中选择候选内容，再按任务、阶段和关键特征进行分组，随后合成为可广播的全局指导载荷，最后回传给边缘侧相应节点。

最后，从通道数据流看，图 3-2 中包含五类关键通道。通道 1 表示轨迹上传，即边缘客户端将阶段轨迹发送至云端服务；通道 2 表示总结请求与响应，即边缘侧在缓存未命中时请求云端生成总结，并接收云端返回的摘要结果；通道 3 表示全局指导注入，即云端聚合器将整理后的全局指导反馈到边缘客户端；通道 4 表示记忆注入，即边缘侧将用户任务、上下文示例和工作记忆注入智能体执行过程；通道 5 表示确认回执，即边缘侧对接收到的全局指导进行确认。通过这些通道，图中的边缘执行、云端评估、记忆沉淀和反馈回流形成了完整的数据闭环。

3.2 边缘侧执行机制与本地记忆设计

3.2.1 子目标驱动的边缘执行流程

边缘侧的首要目标是保证任务执行过程中的低时延与高连续性。为此，本文在边缘节点中采用面向子目标的执行方式，将长时程任务拆分为若干具有阶段性的局部目标，并围绕当前子目标组织动作生成与环境交互。ReAct 和 HiAgent 等研究说明，复杂智能体任务通常需要在多轮推理、行动和阶段化上下文管理中逐步推进[21,25]；据此，子目标驱动的执行模式能够把长链任务转化为若干相对清晰的阶段，从而降低单次决策所需处理的上下文规模，提高边缘模型在资源受限条件下的稳定性。

在实际执行过程中，边缘智能体维持一个持续滚动的本地决策闭环。系统首先根据当前任务状态确定待完成的子目标，然后基于当前观测、本地记忆和已有执行历史生成下一步动作；执行动作后，环境返回新的观测信息，边缘侧据此更新本地状态并进入下一轮推理。上述规划、动作选择和环境交互均在边缘节点本地完成，因此能够较好满足实时任务对响应速度的要求。即使云端暂时不可用，边缘智能体仍可依赖本地机制继续推进当前任务。

结合式 (2-1) 中对智能体动作生成的抽象，本文在边缘侧将其中的历史信息具体收束为本地记忆 M_t^{edge}，并以当前子目标 g_t^{sub} 作为动作选择的阶段约束。也就是说，边缘决策始终围绕当前观测、当前子目标和本地可用记忆展开，而不是无差别地处理完整全局历史。

子目标驱动的另一个优势在于，它为长程任务提供了更清晰的过程边界。在块世界、抓手搬运、文本交互等任务中，智能体常常需要经历多个相互依赖的阶段，如果始终以单一全局目标组织执行过程，容易导致当前步骤与历史信息混杂在一起，进而增加

推理噪声。将任务拆分为若干局部阶段后，边缘智能体能够更清楚地判断当前应关注的内容、已经完成的内容以及下一阶段需要衔接的内容，这对于长任务下的连续推进尤为重要。

此外，本文的边缘执行流程强调对失败信息的即时吸收。当某一步动作未达到预期效果时，边缘侧并不立即依赖云端介入，而是先结合当前观测与本地历史进行自我修正。这意味着云边协同并不是对边缘自主性的替代，而是在边缘本地执行能力基础上的增强。只有在阶段性轨迹形成后，云端的总结与知识整理才开始发挥更明显的作用。

从输入输出关系看，边缘侧在每一轮主要接收三类信息：当前环境观测、当前子目标以及本地记忆中与该子目标相关的上下文；输出则包括当前动作以及阶段结束后形成的阶段轨迹。前者直接作用于环境交互，后者则作为后续摘要和云端处理的基础。因此，边缘侧既承担在线执行功能，也负责为后续经验整理提供原始阶段样本。

3.2.2 本地分层记忆组织方式

为了避免长时程任务中的历史轨迹持续膨胀，本文在边缘侧采用分层记忆组织方式。已有智能体记忆研究强调，需要区分当前上下文维护与长期经验保存，并通过选择、压缩和检索提高记忆利用效率[17,19]。据此，本文对于当前正在执行的子目标，保留较为细粒度的动作与观测记录作为工作记忆；对于已经完成的子目标，则仅保留压缩后的阶段摘要。这样，边缘节点在生成下一步动作时主要使用与当前阶段直接相关的局部信息，而不必反复携带完整历史。

为便于表述，本文将边缘记忆记为

$$M_t^{\text{edge}} = \left\{ M_t^{\text{cur}}, M_t^{\text{his}} \right\}, \tag{3-1}$$

式中 M_t^{cur} ——当前子目标对应的工作记忆；

M_t^{his} ——已完成子目标的阶段摘要集合。

这种设计与前文讨论的工作记忆和长期经验相对应。当前阶段的细粒度轨迹承担工作记忆的作用，服务即时决策；已完成阶段的阶段摘要则保留任务推进脉络，并作为后续形成长期经验的基础。二者分开维护后，边缘侧在构造上下文时可以优先保留当前阶段的必要细节，同时用较短的摘要表示更早阶段的执行结果。式 (3-1) 的作用不在于严格划分记忆边界，而在于明确不同层级信息在决策中的不同地位：前者更偏向即时响应，后者更偏向阶段衔接与历史参考。

3.2.3 轨迹摘要与本地轻量化维护

在该记忆结构之上，边缘侧还对每个已完成子目标生成阶段摘要，用于替代原始轨迹进入历史记忆。该摘要至少保留子目标、关键状态变化、主要动作以及阶段结果，从

而使边缘侧能够在较短上下文内回顾阶段执行内容、结果以及后续是否需要规避类似错误。

从方法上看，摘要操作相当于把原始局部轨迹 $\hat{\tau}_j^i$ 映射为更紧凑的阶段表示 s_j^i，即

$$s_j^i = \mathrm{Summarize}\left(\hat{\tau}_j^i\right), \tag{3-2}$$

式中 s_j^i ——第 j 个子目标阶段对应的摘要表示；

$\mathrm{Summarize}(\cdot)$ ——轨迹摘要映射过程。

阶段摘要同时也是上传到云端的基本输入单元。相比直接传输未经整理的长轨迹，摘要化表示更适合后续的云端分析与知识归档，也更容易在不同边缘节点之间形成可比较的经验单元。因此，边缘侧的摘要过程既服务本地轻量化维护，也为云端反馈提供了统一入口。

需要说明的是，边缘侧上传的并不是孤立的一段文本摘要，而是阶段摘要、任务标识和子目标标识组成的组合信息。这样做的原因在于，单独的摘要文本只能描述局部执行经过，而无法说明该阶段在整项任务中的位置；加入任务与子目标标识后，云端才能将当前上传内容与同类历史经验对应起来，并进一步决定应返回局部反馈还是检索全局指导。通过这一处理，边缘侧无需在后续轮次反复携带完整原始轨迹，而可以在保留关键信息的同时控制上下文长度。

3.3 云端评估机制与知识管理设计

3.3.1 云端评估服务流程

云端部分负责接收边缘侧上传的阶段记录，并将其转化为后续可复用的反馈信息和长期经验。结合前文的问题设定，本文将云端反馈分为两条路径：一条是面向单个已完成阶段的单轨迹反馈，另一条是基于多节点历史经验形成的全局指导。前者借鉴了利用轨迹反馈改进行为决策的思路[22]，后者则对应跨任务经验检索与复用的经验归纳思路[29]。

在具体流程上，云端处理的基本单元不是完整任务历史，而是边缘侧上传的阶段摘要及其对应的任务标识信息。这样的设计使云端能够围绕单个已完成阶段开展评估，而不必等待整项任务结束后再统一分析。对于长时程任务而言，只有将阶段经验在任务进行过程中及时整理，前面阶段形成的信息才可能对后续阶段产生实际作用。

围绕这一阶段记录，云端首先生成与当前节点直接相关的单轨迹反馈，包括阶段完成情况、关键失误、有效策略以及对下一相邻阶段的简要建议；随后，再判断其中哪些内容适合被长期保留并进入知识归档。若将云端针对第 j 个阶段输出的结果记为 R_j^i，则可表示为

$$R_j^i = \left\{c_{j,\mathrm{local}}^i,\ e_j^i\right\}, \tag{3-3}$$

式中 $c^i_{j,\mathrm{local}}$ ——回传给当前节点的单轨迹反馈；

e^i_j ——从当前阶段提取的长期经验候选。

式 (3-3) 表明，云端评估并不只输出面向当前任务的即时反馈，还会同步产生后续知识归档所需的经验候选。

其中，$c^i_{j,\mathrm{local}}$ 强调时效性，主要服务当前任务的后续阶段；e^i_j 则强调可复用性，作为后续长期经验归档与经验汇聚的基础。前者说明上一阶段发生了什么以及下一阶段应如何调整，后者说明这一阶段积累了哪些值得保留的经验。因此，云端评估并不只是对执行结果的事后说明，而是同时承担了反馈生成与经验沉淀的作用。

当云端积累了来自多个节点、多个轮次的相似阶段记录后，还会进一步整理其中重复出现的有效模式和常见失败原因，形成更稳定的全局指导。与单轨迹反馈相比，这类信息不再严格绑定某一次具体执行，而是面向相近任务阶段提供较一般化的经验提示。换言之，云端评估首先说明上一阶段发生了什么，后续归档与汇聚则说明类似阶段中哪些经验值得重复利用。

3.3.2 云端知识存储结构

为了支持上述两类反馈，本文在云端采用短期缓存与长期经验组成的两层存储结构。若将云端存储整体记为 M^{cloud}，则其可表示为

$$M^{\mathrm{cloud}} = \left\{M^{\mathrm{stm}}, M^{\mathrm{ltm}}\right\}, \tag{3-4}$$

式中 M^{stm} ——短期缓存；

M^{ltm} ——长期经验。

其中，短期缓存用于保存最近上传的阶段记录及其反馈结果，使云端能够快速完成当前轮次的反馈生成；长期经验用于保存经过整理后的经验条目，支撑后续的检索与跨节点复用。前者面向最近一次或最近几次交互，服务当前任务过程中的及时回流；后者面向跨轮次积累，服务后续任务中的经验复用。通过这种分层方式，云端不必把所有上传内容都直接视为稳定经验，而是先完成近期反馈，再逐步沉淀可重复利用的长期经验。

从过程上看，云端对一个阶段记录的处理可概括为接收、暂存、评估和归档四步。算法 1 给出了云端侧评估与知识更新的主要步骤：阶段记录首先进入短期缓存 M^{stm}，随后生成面向当前节点的单轨迹反馈 $c^i_{j,\mathrm{local}}$；若该阶段记录具有复用价值，则进一步沉淀到长期经验 M^{ltm} 中，并在相似经验达到条件时形成全局指导 $c^i_{j,\mathrm{global}}$。

算法 1 云端评估与知识更新流程

输入: 阶段摘要 s_j^i，阶段标识信息 m_j^i，短期缓存 M^{stm}，长期经验 M^{ltm}

输出: 回传反馈 C_j^i，更新后的短期缓存 M^{stm}，更新后的长期经验 M^{ltm}

将当前阶段记录写入短期缓存 M^{stm}
针对当前阶段生成单轨迹反馈 $c_{j,\text{local}}^i$
初始化回传反馈 $C_j^i \leftarrow \{c_{j,\text{local}}^i\}$
从当前阶段记录中提取可复用经验候选
if 当前经验满足归档条件 **then**
    写入长期经验 M^{ltm}
end if
if M^{ltm} 中存在足够的相似阶段经验 **then**
    汇聚相似经验并形成全局指导 $c_{j,\text{global}}^i$
    更新回传反馈 $C_j^i \leftarrow C_j^i \cup \{c_{j,\text{global}}^i\}$
    更新长期经验 M^{ltm}
end if
return C_j^i，M^{stm} 和 M^{ltm}.

3.3.3 知识归档与经验沉淀

在知识归档阶段，云端并不直接保存原始长轨迹，而是先将经验条目组织为更稳定、更可检索的形式。智能体综述和生命周期学习研究均强调，长期经验需要经过结构化管理，才能在后续任务中发挥持续作用[26,51]。据此，本文进一步围绕任务类型、任务变体和阶段目标组织云端经验条目。若将长期经验进一步划分，则可写为

$$M^{\text{ltm}} = M^{\text{local}} \cup M^{\text{global}}, \tag{3-5}$$

式中 M^{local} ——来源于单次阶段执行的单阶段经验集合；

M^{global} ——由多条相近经验进一步整理得到的全局指导集合。

其中，M^{local} 主要保留某一阶段中有效做法和易失败做法等与单次执行直接相关的信息，用于支持后续的相似阶段检索；M^{global} 则对应更稳定的执行规律，用于描述跨节点、跨轮次重复出现的经验模式。前者强调具体性，后者强调可迁移性，二者共同构成云端长期经验的主要内容。

随着边缘节点数量和执行轮次增加，云端积累的已归档经验会逐步覆盖更多任务阶段。这意味着某一节点在一次任务中形成的有效经验，可以在后续轮次甚至其他节点中被再次利用。对于难以频繁更新模型参数的边缘场景而言，这种依靠外部知识持续改进的方法更符合实际部署条件。

3.4 云边协同机制实现

3.4.1 轨迹上传与结果回传流程

云边协同的关键在于确定何时上传轨迹、何时使用反馈。本文选择在边缘节点完成一个子目标后触发上传，而不是在每一步动作之后都请求云端分析。这样可以保证单次上传携带较完整的阶段语义，同时避免过高的通信频率打断边缘侧连续执行。

在一次上传过程中，边缘侧提交的是当前阶段的摘要轨迹及其任务标识信息；云端完成处理后返回两类结果：与该阶段直接相关的单轨迹反馈，以及在条件满足时可供同类任务阶段使用的全局指导。前者优先服务当前节点的后续子目标，后者则可继续写入长期经验，并在其他相似节点上复用。

3.4.2 反馈信息注入与本地更新

为了保证边缘执行的连续性，云端回传的信息不会直接替代边缘侧当前状态，而是作为附加上下文注入下一轮决策。边缘节点仍以当前观测和本地记忆为主，云端反馈则提供与已有经验相关的补充信息，例如上一阶段的失误提示、对下一阶段的简要建议，或来自其他节点的通用经验。

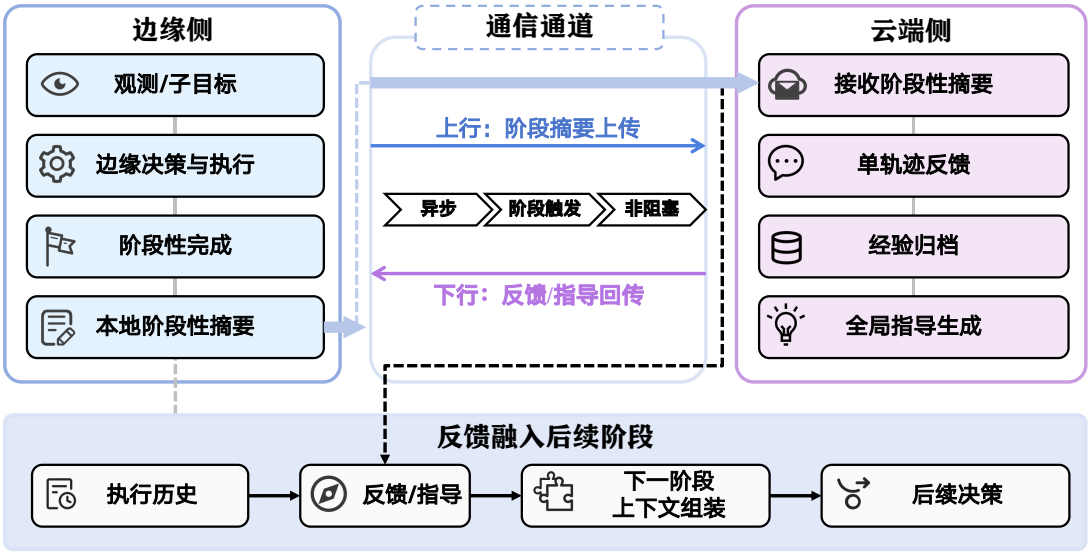


图 3-3 云边协同执行与反馈回流流程图

算法 2 云边执行与反馈更新流程

输入: 当前观测 o_t^i，当前子目标 g_t^{sub}，边缘本地记忆 M_t^{edge}，云端长期经验 M^{ltm}

输出: 可用回传反馈 C_t^i，下一步决策上下文 P_{t+1}^i，更新后的边缘本地记忆 M_{t+1}^{edge}

初始化回传反馈 $C_t^i \leftarrow \varnothing$
for 每个决策步 t **do**
  生成动作 $a_t \leftarrow \pi(o_t^i, g_t^{\text{sub}}, M_t^{\text{edge}})$
  执行动作 a_t 并获得新观测 o_{t+1}^i
  if 当前子目标已完成 **then**
    构造阶段摘要 $s_j^i \leftarrow \text{Summarize}(\hat{\tau}_j^i)$
    上传 $(s_j^i, \textit{task id}, \textit{subgoal id})$ 至云端
    云端生成单轨迹反馈 $c_{j,\text{local}}^i$
    设置回传反馈 $C_t^i \leftarrow \{c_{j,\text{local}}^i\}$
    if M^{ltm} 中存在足够的相似经验 **then**
      生成全局指导 $c_{j,\text{global}}^i$
      更新回传反馈 $C_t^i \leftarrow C_t^i \cup \{c_{j,\text{global}}^i\}$
    end if
    将可复用经验写入云端长期经验
    将可用反馈返回边缘智能体
  end if
  组装下一步上下文 $P_{t+1}^i \leftarrow \mathscr{A}(o_{t+1}^i, g_{t+1}^{\text{sub}}, M_t^{\text{edge}}, C_t^i)$
  更新当前子目标和本地工作记忆
end for
return C_t^i，P_{t+1}^i 和 M_{t+1}^{edge}.

若将云端反馈记为 C_t^i，则边缘在下一轮决策中使用的上下文可写为

$$P_{t+1}^i = \mathscr{A}\left(o_{t+1}^i, g_{t+1}^{\text{sub}}, M_t^{\text{edge}}, C_t^i\right), \tag{3-6}$$

式中 P_{t+1}^i ——节点 i 在时刻 $t+1$ 的决策上下文；

g_{t+1}^{sub} ——节点 i 在时刻 $t+1$ 需要处理的当前子目标；

C_t^i ——云端在时刻 t 回流的反馈信息；

$\mathscr{A}(\cdot)$ ——上下文组装过程。

因此，边缘节点在接收到反馈后，主要做的是更新下一轮上下文，而不是重写已经完成的阶段。式 (3-6) 说明云端反馈并不是独立替代边缘状态的信息源，而是作为附加上下文被整合进下一轮决策输入。这里的 C_t^i 既可以表示针对上一阶段轨迹生成的单轨迹反馈，也可以包含从长期知识中检索得到的全局指导。

图 3-3 给出了反馈回流在协同过程中的位置。边缘侧在完成阶段执行后上传摘要，

云端形成单轨迹反馈并在必要时结合长期经验生成全局指导，随后这些结果被回传到边缘侧，参与下一阶段的上下文组装。算法 2 进一步给出了反馈注入与本地更新在一次协同执行中的主要顺序。

进一步地，本地更新并不是把云端返回内容原样拼接到提示中，而是根据当前子目标选择性注入相关反馈。若当前任务仍处于相近阶段，则更侧重使用单轨迹反馈；若进入新的但结构相似的阶段，则更侧重引用全局指导。这样可以避免无关反馈挤占有限上下文，也使云端信息的使用方式与当前决策阶段保持一致。

3.4.3 跨边缘经验共享

除了面向单个节点的单轨迹反馈外，本文还利用云端完成跨边缘经验共享。这里的‘‘跨’’并不是指边缘节点之间直接交换轨迹，而是指多个边缘节点将阶段记录上传到同一云端知识空间后，由云端统一完成归档、分组、汇聚和再分发。具体而言，边缘侧上传的记录会携带任务类型、任务变体、节点标识、子目标文本以及规范化后的子目标标识等元数据；云端在完成单轨迹评估后，将满足归档条件的经验写入长期知识库，并保留其来源节点和阶段语义。

在实现上，云端首先使用短期缓存保存最新上传轨迹，并在评估完成后把高质量阶段经验写入向量化长期知识库。随后，聚合器（Aggregator）围绕任务类型、任务变体、经验来源命名空间和子目标标识对经验进行分组：当若干经验属于同一类阶段时，云端生成更稳定的全局指导；当精确子目标条件不足时，则按较粗粒度的任务变体进行经验汇聚。该过程使来自不同边缘节点的阶段经验不再只服务于原节点，而是能够被整理为后续节点可检索的知识条目。

若将节点 i 当前阶段的标识信息记为 m_j^i，则云端为该节点检索到的全局指导可表示为

$$c_{j,\mathrm{global}}^{i} = \mathrm{Select}\left(m_j^i, M^{\mathrm{global}}\right), \tag{3-7}$$

式中 $c_{j,\mathrm{global}}^{i}$ ——节点 i 在第 j 个阶段可用的全局指导；

m_j^i ——当前阶段对应的任务类型、任务变体和子目标标识等元数据；

$\mathrm{Select}(\cdot)$ ——根据阶段标识从全局经验中选择相关指导的过程。

式 (3-7) 强调，跨边缘共享依赖的是阶段语义和任务命名空间的匹配，而不是简单复用另一节点的完整执行轨迹。

需要强调的是，跨边缘共享的对象并不是其他节点的原始轨迹，而是经云端整理后的压缩经验。完整轨迹通常包含较强的实例依赖性，直接回传给其他节点容易引入无关细节；相比之下，经过筛选的全局指导更适合作为附加上下文进入后续决策过程。当前边缘节点请求指导时，云端会通过检索与匹配机制选择与当前阶段兼容的经验，并将其以统一的全局指导（Global Guidance）形式返回边缘侧，而不会替代最新环境观测或直

接覆盖本地决策。

这种共享机制尤其适用于边缘场景，因为单个节点通常只能接触有限任务样本，难以仅依靠自身历史积累足够稳定的经验。通过云端的归档、分组和选择，不同节点形成的单阶段经验可以转化为跨节点流动的长期经验，从而提升整体经验利用效率。

3.4.4 协同闭环的讨论

综上，本文的云边协同机制可以概括为边缘执行当前阶段、云端整理已完成阶段、反馈再次进入后续阶段。这一闭环之所以能够成立，主要依赖三个条件：

(1) 边缘侧按子目标组织执行过程，使长任务被划分为可独立处理的阶段单元；

(2) 云端按阶段接收和整理轨迹，而不是等待完整任务结束后再统一处理；

(3) 云端返回的信息不接管当前动作生成，而是在下一阶段作为附加上下文参与决策。

在这三个条件下，边缘在线执行与云端异步经验整理可以形成可追踪的衔接关系。

从闭环运行方式看，边缘侧负责维持当前任务的连续推进，云端侧负责将已完成阶段转化为单轨迹反馈和全局指导，随后这些信息再进入边缘侧下一阶段的上下文组装过程。这样，前一阶段形成的经验可以在任务尚未结束时就影响后一阶段的决策，而不必等到整轮任务完成后再进行总结。

对于本文关注的长时程任务，这种闭环的直接作用并不在于让云端替代边缘决策，而在于让边缘侧在保持轻量执行的同时获得来自长期经验的补充支持。也就是说，边缘侧工作记忆仍然是动作生成的主干输入，云端返回的单轨迹反馈与全局指导只作为附加上下文参与后续阶段的提示组装，并不直接接管当前动作生成过程。本文方法改变的不是边缘节点的基本决策入口，而是后续阶段可利用的上下文内容，从而在不显著增加边缘负担的前提下提升后续决策质量。

3.5 可视化支撑模块设计

3.5.1 可视化支撑模块整体架构

对于本文所研究的云边协同长任务处理方法，仅报告最终成功率、平均步数或上下文开销，并不足以完整说明协同机制是否真正发挥了作用。许多关键问题都发生在任务推进过程中，例如边缘节点是否持续在线、阶段轨迹是否及时上传、云端反馈是否顺利回流，以及不同节点之间是否形成了稳定的经验共享。为此，本文在方法原型之外进一步引入一个面向实验观测的可视化支撑模块，用于对协同链路中的关键状态进行集中展示。

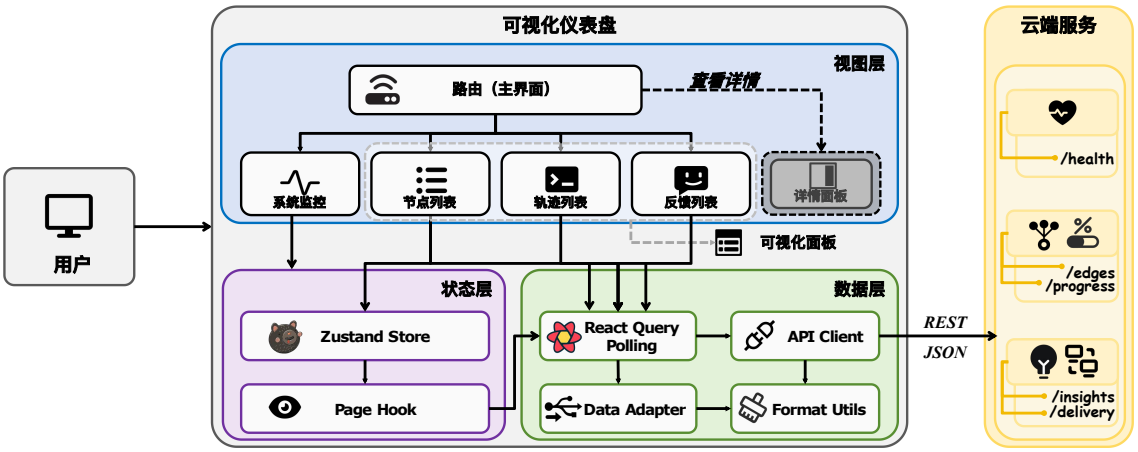


图 3-4 可视化支撑模块整体架构示意图

该模块不进入边缘侧动作生成回路，而是在执行链路之外读取已经产生的运行状态，并将边缘执行、轨迹上传、云端反馈与结果回流等分散信息组织为统一的实验视图。这样，实验分析不只依赖最终结果表格，也可以对应到任务推进过程中的具体环节。

图 3-4 给出了该可视化支撑模块的整体架构。模块由展示层、状态层、数据层和服务接口层构成。展示层对应前端页面中的系统总览、节点状态、轨迹事件流、全局指导和详情抽屉等区域；状态层维护当前选中节点、任务过滤、自动滚动、暂停刷新和详情面板等交互状态；数据层通过轮询调度、接口请求封装和视图模型适配，将后端记录转换为页面可直接使用的数据对象；服务接口层由云端提供健康状态、边缘节点进度、单节点详情和全局指导投递状态等观测入口。通过这种分层组织，该模块能够在不改变原有协同逻辑的前提下持续重组实验过程中的关键状态。

3.5.2 可视化支撑模块功能架构

从观测对象看，本文的可视化支撑模块主要围绕三类信息展开。第一类是边缘侧任务状态，包括节点连接情况、任务进度、所处阶段及阶段切换，用于反映边缘执行是否稳定推进。第二类是轨迹事件流信息，包括任务开始、阶段完成与反馈应用等过程事件，用于呈现长任务在时间维度上的推进脉络。第三类是云端反馈信息，包括全局指导生成情况、反馈确认状态和应用状态，用于观察经验回流链路是否真正闭合。对于多节点并发场景，这些信息还能够帮助研究者比较不同节点的推进节奏，判断性能变化究竟来自任务本身，还是来自协同链路中的某一环节。

图 3-5 可视化支撑模块功能架构示意图

图 3-5 从功能组织角度说明了该模块的信息结构。界面围绕状态总览、监控面板、详情分析和运行控制四类功能展开。状态总览提供云端健康状态、边端连接数量和最近同步时间；监控面板同时展示节点状态、轨迹事件流和全局指导；详情分析用于查看节点、事件和指导记录的结构化内容及原始载荷；运行控制则负责节点选择、任务过滤、自动滚动和暂停刷新等交互。由此，该模块能够沿着边缘执行、轨迹上传、云端反馈和结果回流的链路连续呈现实验状态。下一节将进一步说明该可视化支撑模块的具体实现方式。

3.6 可视化支撑模块实现

在完成整体架构与功能架构设计后，本节进一步说明可视化支撑模块的具体实现。该模块实现可以分别从技术栈与功能栈两个层面进行概括：前者强调前后端运行环境、页面组织和数据访问方式，后者强调不同信息区域由哪些前端模块承载。

表 3-1 总结了该模块实现所依赖的关键技术栈。其中，前端部分主要负责页面构建、交互组织与数据映射，后端部分则负责提供云端状态、边缘进度和反馈记录等观测入口。由此，可视化支撑模块得以在不改变原有协同逻辑的前提下，将前文方法中的关键中间结果持续暴露给实验观察过程。

表 3-1 可视化支撑模块技术栈

侧别	层次	关键技术栈	版本	作用
前端	页面框架	React	18.3.1	页面构建与运行承载
		TypeScript	5.6.3	
		Vite	5.4.10	
	前端路由	React Router	6.28.0	页面路由与区域组织
	状态管理	Zustand	5.0.1	交互状态统一维护
	数据调度	React Query	5.59.20	数据轮询与缓存刷新
	接口访问	Fetch API	浏览器内置	接口请求与视图映射
		REST 适配层	自定义	
	界面组件	Radix UI	1.1.x	界面组件与样式呈现
		Tailwind CSS	4.1.4	
后端	云端环境	FastAPI	0.68+	云端服务与状态组织
		Uvicorn	0.15+	
		ChromaDB	0.4+	
		Redis	4.0+	
		WebSockets	15.0.1	
	边缘环境	Gym	0.24.0	边缘通信与任务接入
		Gymnasium	1.1.1	

从后端接入关系看，可视化支撑模块并不是独立生成新数据，而是直接承接前文中定义的云边协同过程。边缘节点按照式 (2-3) 所示的阶段化轨迹形式组织执行历史，并在子目标结束后通过式 (3-2) 将局部轨迹压缩为阶段摘要；随后，云端按照算法 1 和算法 2 完成阶段记录接收、反馈生成、经验归档与结果回传，并进一步按照式 (3-3) 产生单轨迹反馈与长期经验候选。模块后端接口正是围绕这些已有过程建立的：一类接口用于暴露边缘节点的进度与事件状态，一类接口用于暴露云端反馈的生成、确认与应用状态，另一类接口用于反映云端服务的整体健康情况。

在上述技术栈之上，前端界面按照功能责任进一步划分为概览、监控、分析和控制

四类区域。表 3-2 总结了这些功能区域与前端承载模块之间的对应关系，作为后续页面实现说明的总览。

表 3-2 可视化支撑模块功能栈实现

类别	功能层	核心展示内容	前端承载模块	作用
概览	状态总览	云端状态、节点数量 同步时间	SystemHeader	全局态势总览
监控	节点状态	节点列表、连接状态 任务进度、事件摘要	NodeListPanel NodeCard	节点推进对比
	轨迹事件	进度卡片、事件时间线 阶段切换、事件定位	TrajectoryStreamPanel TaskProgressCard TrajectoryEventRow	过程时序追踪
	云端反馈	全局指导、确认状态 应用状态、事件关联	CloudFeedbackPanel InsightCard	反馈回流观察
分析	详情分析	节点详情、事件载荷 指导记录、原始数据	DetailSheet	记录钻取分析
控制	运行控制	节点选择、任务过滤 自动滚动、暂停刷新	状态存储 页面控件	页面联动控制

从前端承载方式看，该模块并不直接渲染云端返回的原始记录，而是先将其转换为适于页面展示的视图对象。节点状态被组织为节点视图，用于承载连接状态、任务信息、运行时长和最近事件；轨迹记录被组织为事件视图，用于承载时间戳、事件类型、阶段推进信息与关联反馈；云端结果则被组织为指导视图，用于承载摘要、作用范围、确认状态和应用状态。通过这种数据承载方式，该模块能够将后端内部记录转化为可比较、可筛选和可联动的前端信息单元。下文将围绕状态总览、监控面板、详情分析和运行控制说明各区域的实现细节。

3.6.1 状态总览

图 3-6 展示了可视化支撑模块在两种主题下的页面效果。该区域集中展示云端健康状态、已连接边端数量、已跟踪节点数量和最近同步快照，使观察者能够先判断云端服务与边缘节点是否处于可观测状态。

在数据来源上，系统头部通过健康检查接口读取云端状态，通过边缘进度接口获取节点统计信息，并将同步时间转换为相对时间与快照时间两类显示形式。页面同时提供日间模式与夜间模式切换，主题状态保存在浏览器本地，并通过根节点的样式类控制整体界面风格。

(a) 日间模式

(b) 夜间模式

图 3-6 可视化支撑模块总览与主题切换界面

3.6.2 监控面板

主界面的监控面板由节点状态、轨迹事件流和全局指导三部分构成，如图 3-6 所示。三类面板分别对应边缘节点、任务事件和云端反馈，使实验过程能够从节点、事件和反馈三个角度被连续观察。

节点状态区域展示各边缘节点的连接状态、任务信息、样例进度、运行时长和最近事件摘要；选中节点后，轨迹事件流随之聚焦该节点的任务推进过程。**轨迹事件流区域**提供任务筛选、事件时间线、阶段信息、自动滚动和暂停刷新等观察入口，并结合实时轮询、任务执行和阶段扫描提示呈现边缘侧运行状态。**全局指导区域**展示云端下发的全局指导记录，包括指导摘要、作用范围、目标边端、确认状态和应用状态；事件流中的指导注入与应用事件则作为辅助观察信息，用于定位反馈回流在运行过程中的具体位置。

3.6.3 详情分析

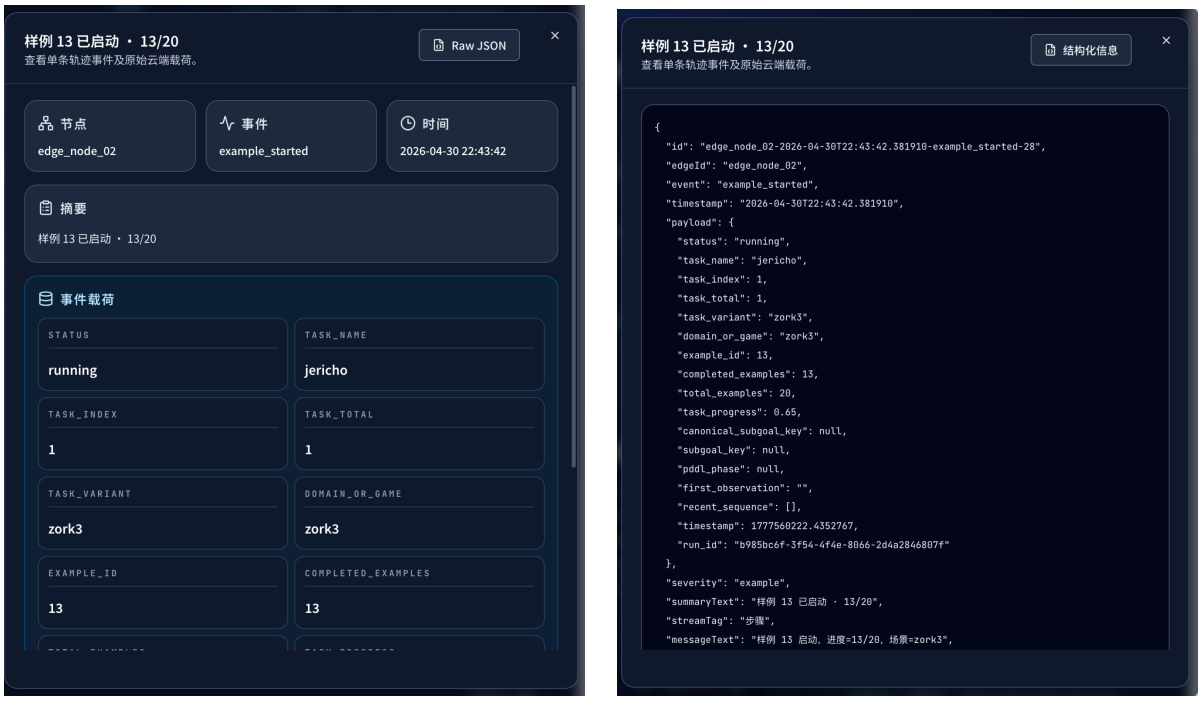
详情分析通过右侧抽屉展开，用于查看节点、轨迹事件和全局指导三类对象的细粒度记录。该抽屉保留结构化信息与 Raw JSON 两种视图：前者服务快速阅读，后者用于核对后端载荷字段。

- 轨迹事件详情展示事件摘要、事件时间、所属节点以及后端载荷字段，如图 3-7 所示。该视图用于将事件流中的一条记录展开到具体载荷，便于检查阶段切换、反馈应用或异常事件是否与后端记录一致。
- 边缘节点详情展示节点连接状态、最近更新时间、任务状态字段和近期事件，如图 3-8 所示。该视图用于判断某一节点当前处于运行、断开、完成或异常等状态，并将节点级状态与近期事件联系起来。
- 全局指导详情展示指导摘要、确认与应用数量、下发范围、阶段标识、来源轨迹和目标边端，如图 3-9 所示。该视图用于检查全局指导是否已经被目标节点确认或应用，并对应到前文所述反馈回流链路。

3.6.4 运行控制

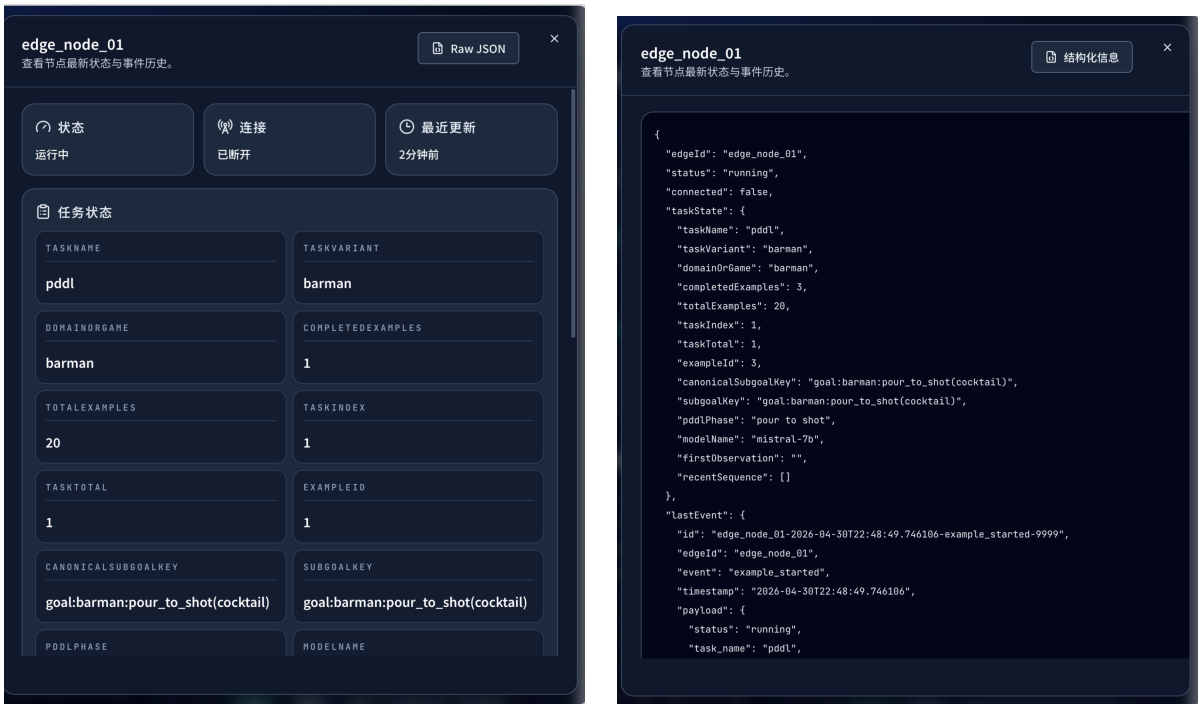
运行控制负责调节页面观察节奏，而不改变边缘智能体的动作生成逻辑。前端主要通过 HTTP 接口向云端请求四类信息：云端健康状态、边缘节点进度信息、单节点详情信息以及全局指导投递信息。

页面使用定时轮询刷新这些数据，并根据浏览器页面可见状态动态调整刷新间隔；当用户暂停刷新时，节点进度、指导投递和健康检查停止自动更新。与此同时，页面状



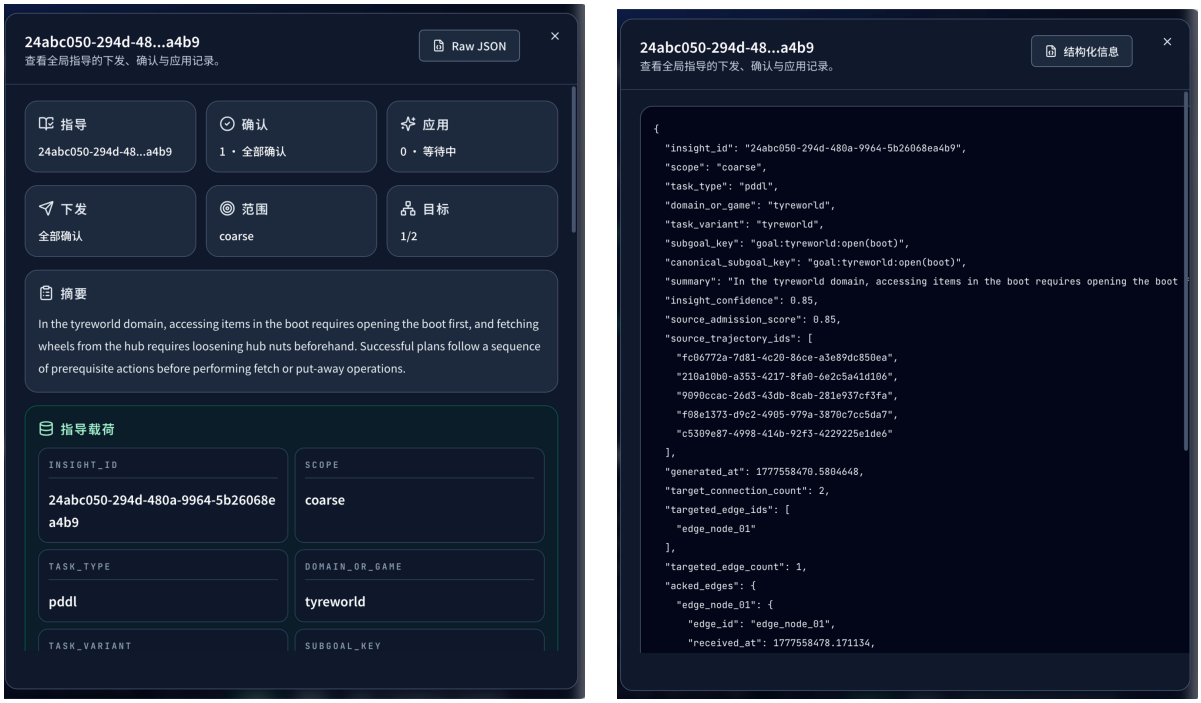
(a) 结构化信息 (b) 原始载荷

图 3-7 轨迹事件详情展示



(a) 结构化信息 (b) 原始载荷

图 3-8 边缘节点详情展示



(a) 结构化信息 (b) 原始载荷

图 3-9 全局指导详情展示

态存储统一维护当前选中节点、任务过滤条件、自动滚动、详情面板和高亮指导记录，使节点列表、事件流、指导面板与详情抽屉能够围绕同一组数据联动更新。

3.7 本章小结

本章围绕资源受限边缘场景中的长任务处理需求，给出了云边协同长任务处理方法的设计与实现思路。具体而言，本章明确了边缘执行、本地记忆、云端反馈与可视化支撑之间的职责划分，设计了以子目标为边界的边缘执行、阶段摘要、云端整理和反馈回流过程，并进一步说明了本地分层记忆、云端知识管理、反馈注入和跨边缘经验共享机制。最后，本章介绍了面向实验监测的 Web 可视化支撑模块，使方法运行过程能够通过节点状态、事件流和反馈回流被持续观察，为后续实验验证提供了基础。

4 实验设计与结果分析

4.1 实验设计

4.1.1 实验目标

本章实验围绕前文提出的云边协同长任务处理方法展开，重点考察在不对边缘侧模型参数进行微调的前提下，云端异步总结、经验整理与反馈回流机制对弱模型边缘节点长时程任务执行能力的影响。场景一以 HiAgent 式本地弱模型执行设置作为对照，并沿用 AgentBoard 评测口径验证云端经验回流机制的直接增益；场景二以场景一中的弱模型边缘节点作为参照，进一步观察强节点经验进入云端记忆后是否带来额外提升。两个场景均采用相同的边缘执行骨架、任务步数上限和记忆预算，仅调整云边协同配置与经验来源方式，从而使实验结果能够更直接地反映云端经验回流机制对弱模型边缘节点任务表现的影响。在此基础上，本文进一步通过关闭云端反馈的纯本地变体开展消融实验，用于区分云端经验回流机制与强节点经验来源对弱模型边缘节点表现的具体作用。

本章结果展示与分析均围绕弱模型边缘节点展开，重点观察其在引入云端反馈后的任务完成情况、任务推进程度、执行开销与动作落地质量，并据此验证方法设计部分所提出方案的有效性。

4.1.2 实验场景

参考 AgentBoard 与长时程智能体相关研究中的任务组织方式[23,25]，本文设置两个实验场景，并在两个场景中始终保持弱模型能力提升这一比较主线。

AgentBoard 是面向大语言模型智能体的综合评测框架，提供了规划、文本交互等多类任务环境，并采用成功率、任务推进率和动作落地准确率刻画智能体在交互过程中的任务完成与动作执行表现[23]。

本文的边缘执行框架主要依照 HiAgent 的长任务执行组织方式构建：HiAgent 本身在 AgentBoard 的任务环境与评价口径基础上开展实验，并进一步引入平均执行步数和上下文消耗，用于分析分层记忆与上下文压缩机制带来的执行开销变化[25]。因此，本文实验中 AgentBoard 主要承担底层任务环境和任务完成类评价指标来源的作用，HiAgent 则提供边缘侧长任务执行框架以及执行开销类评价指标的主要参照。

(1) 场景一：弱模型边缘节点与云端反馈协同

场景一用于验证在不引入强模型边缘节点的条件下，云端经验回流机制对边缘智能体的增益作用。在该场景中，边缘侧采用 Mistral-7B-Instruct-v0.2 作为本地执行模型。

Mistral 7B 是 Mistral AI 提出的 7B 参数开源语言模型，本文在实验中采用其指令微调版本 Mistral-7B-Instruct-v0.2[56]。云端侧采用 deepseek-reasoner[1]作为轨迹总结与反馈生成服务。边缘节点在执行过程中持续进行观测、规划与动作选择，云端则对阶段性轨迹进行异步整理，并将总结后的经验信息回流到边缘侧后续提示中。

该场景下的对照方法为 HiAgent 式本地弱模型执行设置，即在相同任务环境与评价口径下不使用本文设计的云端经验回流机制，仅依赖本地历史与当前观测进行决策。通过这一比较，可以直接观察在不扩大边缘模型规模的情况下，云端异步反馈是否能够改善弱模型的长任务执行表现。

(2) 场景二：强弱协同环境下的弱模型边缘节点

场景二用于进一步验证跨节点经验共享的作用。在该场景中，云端模型采用 deepseek-chat，边缘侧同时存在强模型节点与弱模型边缘节点，其中强模型节点采用 GPT-4 系列模型[57]，实验中实际调用为 gpt-4-turbo[2]，弱模型边缘节点仍采用 Mistral-7B-Instruct-v0.2。实验分析仅关注弱模型边缘节点的表现，并以场景一中的弱模型边缘节点作为参照，不将强模型节点作为主要比较对象。

与场景一不同，场景二中的云端不仅接收弱模型边缘节点的阶段轨迹，也能够吸收强模型节点所产生的高质量执行经验，并将其整理为可复用反馈后提供给弱模型边缘节点。因此，场景二主要考察的问题是：当更高质量的经验被纳入云边协同闭环后，弱模型边缘节点能否在场景一的基础上进一步受益。为突出这一增量效果，后续结果展示以场景一中的弱模型边缘节点表现作为参照。

4.1.3 实验任务与评价指标

(1) 任务设置

本文采用两类长时程任务环境开展实验：PDDL 规划任务与 Jericho 文本交互任务。任务组织方式参考 AgentBoard 与 HiAgent 中对长时程智能体任务的评测设置[23,25]。其中，PDDL 任务来源于国际规划竞赛所使用的 Planning Domain Definition Language 基准集合[58]，本文选取其中具有代表性的 Gripper、Barman、Blocksworld 和 Tyreworld 四个任务，用于考察智能体在不同规划场景中的多步决策能力。与 AgentBoard 和 HiAgent 的实验设定一致，这些任务以文本形式向智能体提供环境观测，使大语言模型能够在自然语言接口下完成规划与执行。

Jericho 是一组建立在虚构世界中的文本交互环境[59]。该任务需要智能体通过探索

[1]本文通过 DeepSeek API 服务调用该模型。

[2]本文通过 OpenAI API 服务调用该模型。

和交互逐步获取世界状态信息，因此对世界建模能力和历史记忆能力具有较高要求。由于原始 Jericho 游戏通常需要较长步骤才能完成，AgentBoard 与 HiAgent 都对目标进行了阶段化改写，使任务能够在有限上下文约束下完成；本文沿用这一设置，将每个冒险任务限制在 15 个子目标以内，以便与前述 PDDL 任务共同构成长时程任务评测环境。

(2) 评价指标

本文采用成功率（Success Rate，SR）、任务推进率（Progress Rate，PR）、动作落地准确率（Grounding Accuracy，GA）、平均执行步数（Steps）和上下文消耗（Context）作为主要评价指标。其中，成功率、任务推进率和动作落地准确率是 AgentBoard 原有评价指标[23]：成功率表示任务最终成功完成的比例；任务推进率表示当前任务状态相对于目标状态的推进程度；动作落地准确率表示智能体输出动作能够被环境正确执行的比例。

平均执行步数和上下文消耗则采用 HiAgent 引入的实验分析指标[25]。其中，Steps 表示智能体在单轮任务中达到任务终止状态前实际执行的平均步数；Context 衡量单轮任务执行过程中提示上下文的累计消耗。

设共有 N 个测试 episode，第 i 个 episode 的实际执行步数为 T_i，第 t 步输入提示的上下文长度为 $|p_{i,t}|$，则某一方法的原始上下文消耗可写为

$$C_{\text{raw}} = \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T_i}|p_{i,t}|. \tag{4-1}$$

式中 C_{raw} ——某一方法在测试任务上的平均原始上下文消耗；

N ——测试 episode 数量；

T_i ——第 i 个 episode 的实际执行步数；

$|p_{i,t}|$ ——第 i 个 episode 在第 t 步输入提示的上下文长度。

设对应参照设置的原始上下文消耗为 C_{ref}，则本文表格中的 Context 以归一化百分比形式报告：

$$\text{Context} = \frac{C_{\text{raw}}}{C_{\text{ref}}} \times 100\%. \tag{4-2}$$

式中 Context ——表格中汇报的归一化上下文消耗；

C_{ref} ——对应参照设置的平均原始上下文消耗。

故 Context 数值越低，表示在相同任务设置下相对于参照方法占用的提示上下文越少；Steps 数值越低，则表示智能体在完成任务或达到终止条件前使用的交互步数更少。

在后续的结果展示中，表 4-2 以 HiAgent 式本地弱模型执行设置作为 C_{ref}，其 Context 记为 100.00%；表 4-3 以场景一中的弱模型边缘节点作为 C_{ref}，其 Context 记为 100.00%；消融实验中的表 4-4 和表 4-5 则以纯本地执行设置作为 C_{ref}，其 Context 记为 100.00%。

对于 PDDL 任务，任务推进率依据当前状态与目标状态之间的匹配分数 r_t^{match} 计算；对于 Jericho 任务，则结合阶段目标完成情况衡量任务推进程度。

4.1.4 实验参数设置

为便于复现实验配置和理解不同场景之间的比较关系，表 4-1 汇总了本文实验中的主要参数设置。

表 4-1 实验参数设置

模块		具体设置	
边缘	执行配置	边缘模型	Mistral-7B-Instruct-v0.2
		决策步数	STEP：30
		记忆预算	Memory Size：100
		推理环境	本地 vLLM
	任务设置	**PDDL**	Blocksworld (10), Gripper (20), Tyreworld (10), Barman (20)
		Jericho	Jericho (20)
云端	云端模型	场景一	deepseek-reasoner
		场景二	deepseek-chat
	云端反馈	*w/ Cloud*	✓
		w/o Cloud	✗
	云端接口	API	DeepSeek API

协同	**场景**	**强节点模型**	**强节点经验**	**Context 参照**
	场景一	–	✗	STANDARD
	场景二	gpt-4-turbo	✓	场景一
	消融实验	–	✗	本地弱模型

注：✓表示启用或使用，✗表示关闭或不使用。

4.1.5 实现细节

在两种实验场景下，边缘侧统一采用方法设计部分实现的执行框架，并保持相同的任务步数上限与记忆预算。本文实现中的边缘执行器为 ContextEfficientAgentV2，其长任务组织方式主要参照 HiAgent。本地记忆预算固定为 100 条历史记录。每个子目标 episode 的最大决策步数设置为 30 步。云端状态上报仅用于任务进度监控，真正参与边缘决策增强的是云端返回的摘要、经验与建议信息。

消融实验沿用相同的弱模型执行器、任务步数上限和记忆预算，仅改变云端反馈是否启用以及云端经验来源。*w/o Cloud* 是本文执行框架内关闭云端反馈后的本地变体，用于刻画云端反馈组件的影响，不等同于表 4-2 中的 HiAgent 式本地对照结果。

上述设置保证了不同实验场景在边缘执行框架、步数上限与记忆预算上的一致性，使实验结果能够更准确地对应到云边协同经验增强机制本身。

4.2 实验结果

4.2.1 场景一

(1) 结果展示

场景一对应的是最直接的验证条件，即在不引入强模型边缘节点的情况下，仅通过云端异步总结和经验回流来提升弱模型的任务表现。表 4-2 给出了该场景的主要结果，其中 Standard 表示 HiAgent 式本地弱模型执行设置下的结果。总体上，该方法相较对照设置取得了更好的任务完成效果，并同时降低了执行成本。

表 4-2 场景一中弱模型边缘节点实验结果

	SR ↑		PR ↑		Steps ↓		Context ↓		GA ↑	
Blocksworld										
STANDARD	0.00		5.00		30.00		100.00%		7.00	
OURS	**20.00**	+20.00	**32.22**	+27.22	**27.98**	-2.02	**68.23%**	-31.77%	**100.00**	+93.00
Gripper										
STANDARD	**0.00**		**4.12**		**30.00**		100.00%		8.67	
OURS	**0.00**	+0.00	2.94	-1.18	**30.00**	+0.00	**59.53%**	-40.47%	**100.00**	+91.33
Tyreworld										
STANDARD	0.00		10.89		30.00		100.00%		20.00	
OURS	**10.00**	+10.00	**32.11**	+21.22	**27.62**	-2.38	**86.38%**	-13.62%	**46.14**	+26.14
Barman										
STANDARD	0.00		0.00		30.00		100.00%		**29.50**	
OURS	**2.50**	+2.50	**6.39**	+6.39	**29.35**	-0.65	**69.60%**	-30.40%	27.08	-2.42
Jericho										
STANDARD	**0.00**		**9.35**		30.00		100.00%		**97.50**	
OURS	**0.00**	+0.00	6.27	-3.08	**29.53**	-0.47	**91.94%**	-8.06%	96.50	-1.00
Overall										
STANDARD	0.00		5.87		30.00		100.00%		32.53	
OURS	**6.50**	+6.50	**15.99**	+10.12	**28.90**	-1.10	**72.12%**	-27.88%	**73.94**	+41.41

具体而言，相较对照设置，成功率提升 6.50 个百分点，任务推进率提升 10.12 个百分点，平均执行步数减少 1.10 步，上下文消耗降低 27.88%，动作落地准确率提升 41.41 个百分点。

上述结果说明，在边缘模型参数保持不变的前提下，云端经验回流已经能够改善弱模型在长时程任务中的整体表现。

(2) 结果分析

场景一呈现出较明显的任务差异性，其增益主要集中在状态依赖较强、阶段衔接更明确的规划类任务上。Blocksworld 和 Tyreworld 的提升最为明显，前者成功率提升 20.00 个百分点，任务推进率提升 27.22 个百分点；后者成功率提升 10.00 个百分点，任务推进率提升 21.22 个百分点。对于这类任务，阶段目标之间的先后关系较清晰，错误动作也更容易在后续阶段被归纳为可复用的经验，因此云端回流的阶段性反馈更容易转化为有效的后续指导。

相比之下，Barman、Gripper 和 Jericho 的收益更具选择性。Barman 虽然在成功率和任务推进率上有所改善，但提升幅度相对有限，这说明其多对象组合和操作约束虽然能从经验总结中获益，但局部反馈未必总能直接转化为稳定的动作落地增益。Gripper 与 Jericho 则未在成功率上取得提升，其中 Gripper 的任务推进率有所下降，Jericho 的任务推进率和动作落地准确率也略有回落。这类结果表明，当任务本身对即时状态判断、交互细节或隐式环境信息更敏感时，云端反馈虽能帮助压缩历史、减少上下文负担，但其对最终任务完成效果的促进并不一定同步显现。

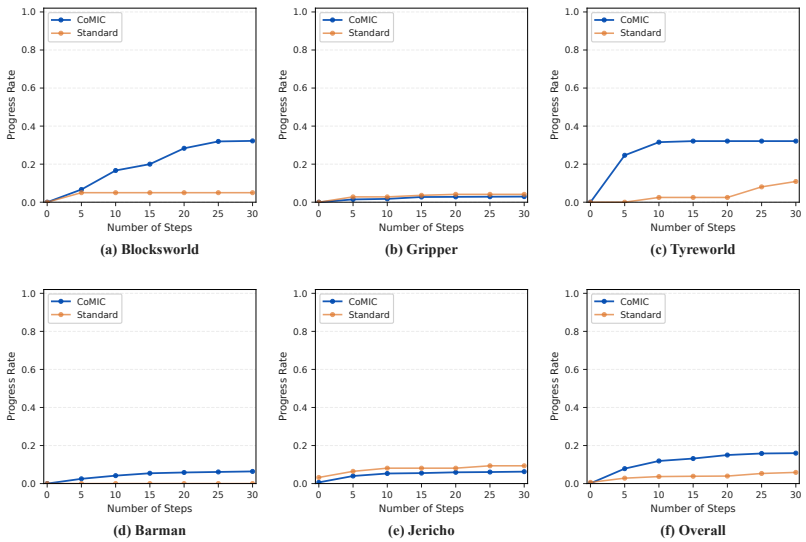


图 4-1 不同执行步上的任务推进率

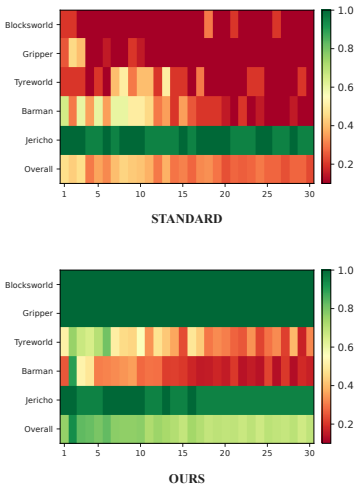


图 4-2 不同执行步上的动作落地准确率

进一步看，场景一中的改进并不是通过增加交互轮次获得的。总体上，平均执行步数减少 1.10 步，Context 降低 27.88%。同时，动作落地准确率提升 41.41 个百分点，在 Blocksworld 和 Gripper 上提升尤为显著。这说明云端反馈的主要作用并不只是延长试错过程，而是在一定程度上改善动作与当前状态之间的匹配关系，并帮助边缘侧在较短上下文内维持更高质量的阶段决策。综合来看，场景一结果表明，云端异步经验回流机制对弱模型长时程任务执行具有总体上的促进作用，但其收益会随任务结构、状态可判别性和经验可迁移程度的不同而表现出明显差异。

图 4-1 和图 4-2 进一步给出了场景一在不同执行步上的动态表现。从整体趋势看，所提方法在多数步数区间内保持了更高的任务推进率和动作落地准确率，这与表 4-2 中成功率、任务推进率和动作落地准确率的提升是一致的，也说明云端回流经验能够在任务执行过程中持续改善边缘节点的决策质量。

4.2.2 场景二

(1) 结果展示

场景二进一步考察强弱协同条件下弱模型边缘节点的表现。与场景一相比，该场景中的云端经验来源包含强模型节点产生的轨迹，因此表 4-3 主要用于观察强节点经验进入云端记忆后，是否能够继续改善弱模型边缘节点的任务表现。需要注意的是，表 4-3 的对比对象不是强模型节点，而是场景一中的弱模型边缘节点；其中 Scenario 1 表示场景一弱模型边缘节点结果，Scenario 2 表示引入强节点经验后的弱模型边缘节点结果。为突出场景间变化，表 4-3 中的 Context 采用相对场景一对应上下文消耗的归一化比例，该处理仅用于展示场景二相对于场景一的增量变化。

总体来看，场景二相较场景一取得了有限但稳定的增益：成功率提升 1.00 个百分点，任务推进率提升 1.79 个百分点，平均执行步数减少 0.13 步，上下文消耗降低 7.73%，动作落地准确率提升 1.15 个百分点。上述结果说明，强节点经验经由云端整理后能够为弱模型边缘节点提供额外帮助，但仍受到弱模型自身规划能力和动作理解能力的限制。

(2) 结果分析

场景二的结果表明，在以场景一弱模型边缘节点为参照时，引入强模型节点轨迹后弱模型边缘节点获得了额外收益，但提升幅度整体较为有限。总体指标中，成功率、任务推进率和动作落地准确率分别提升 1.00、1.79 和 1.15 个百分点，说明强节点经验经过云端整理后能够为弱模型边缘节点提供有效补充；但这些增益并未形成大幅跃迁，表明最终执行仍受限于 Mistral-7B-Instruct-v0.2 的本地规划能力、动作理解能力和对环境反馈的即时判断能力。

从任务维度看，Gripper 和 Jericho 的任务推进率提升较为明显，增幅分别为 3.47 和 5.35 个百分点，说明强节点经验能够帮助弱模型边缘节点在部分任务中更快识别有效状态转移方向。Tyreworld 和 Barman 在成功率上的增幅均为 2.50 个百分点，但其余指标并不完全同步：Tyreworld 的动作落地准确率下降 4.45 个百分点，Barman 的任务推进率下降 1.39 个百分点。这说明强节点经验并非在所有任务中都能被弱模型边缘节点稳定吸收，任务自身的动作约束、状态组合复杂度和目标结构会影响经验迁移效果。

Blocksworld 的变化较小，成功率和动作落地准确率保持不变，任务推进率仅提升

表 4-3 场景二中弱模型边缘节点实验结果

	SR ↑		PR ↑		Steps ↓		Context ↓		GA ↑	
Blocksworld										
SCENARIO 1	20.00		32.22		27.98		100.00%		100.00	
SCENARIO 2	**20.00**	+0.00	**33.33**	+1.11	**27.80**	-0.18	**96.10%**	-3.90%	100.00	+0.00
Gripper										
SCENARIO 1	0.00		2.94		**30.00**		100.00%		100.00	
SCENARIO 2	**0.00**	+0.00	**6.42**	+3.47	**30.00**	+0.00	**97.75%**	-2.25%	100.00	+0.00
Tyreworld										
SCENARIO 1	10.00		32.11		27.62		100.00%		**46.14**	
SCENARIO 2	**12.50**	+2.50	**32.50**	+0.39	**27.25**	-0.37	**91.89%**	-8.11%	41.69	-4.45
Barman										
SCENARIO 1	2.50		**6.39**		29.35		100.00%		27.08	
SCENARIO 2	**5.00**	+2.50	5.00	-1.39	**28.80**	-0.55	**85.29%**	-14.71%	**36.44**	+9.36
Jericho										
SCENARIO 1	0.00		6.27		**29.53**		100.00%		96.50	
SCENARIO 2	**0.00**	+0.00	**11.62**	+5.35	30.00	+0.47	**96.75%**	-3.25%	**97.33**	+0.83
Overall										
SCENARIO 1	6.50		15.99		28.90		100.00%		73.94	
SCENARIO 2	**7.50**	+1.00	**17.77**	+1.79	**28.77**	-0.13	**92.27%**	-7.73%	**75.09**	+1.15

1.11 个百分点。该现象说明，当场景一中弱模型已经通过云端反馈获得较高动作落地质量后，继续引入强节点经验的边际收益会变小。综合来看，场景二并不说明强弱协同能够完全弥补弱模型能力限制，但表 4-3 中 SR、PR 和 GA 的总体提升仍表明，云端经验汇聚与反馈机制能够在已有弱模型能力基础上带来进一步增强。

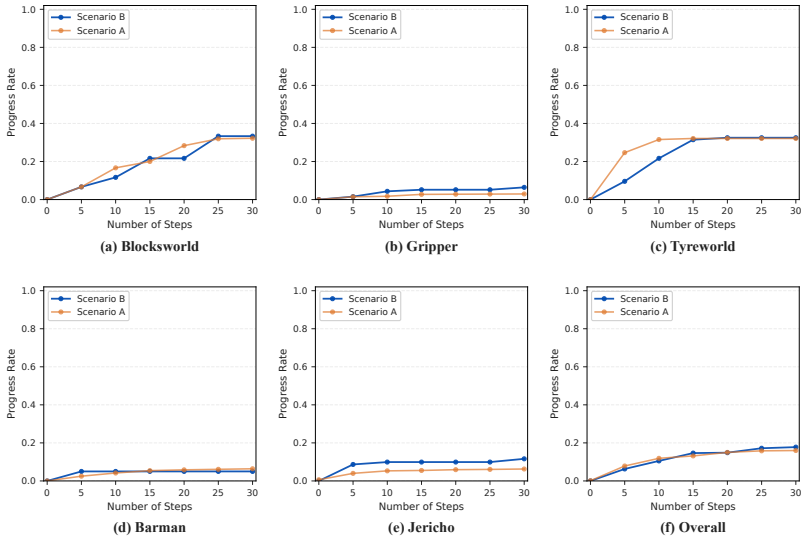


图 4-3 场景二不同执行步上的任务推进率

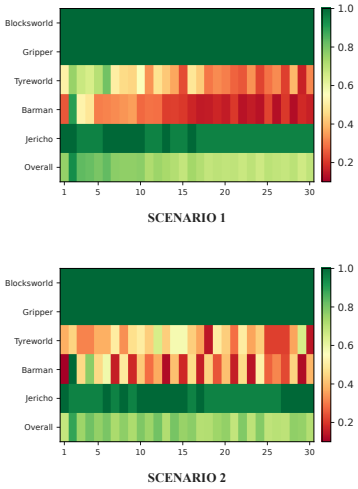


图 4-4 场景二不同执行步上的动作落地准确率

图 4-3 和图 4-4 给出了场景二在不同执行步上的动态表现。与表 4-3 的结果一致，场景二的曲线变化体现出更明显的选择性：部分任务在中后期步骤上获得更高推进率或动作落地准确率，但整体提升不如场景一显著。这进一步说明，强节点经验的作用主要体现在补充和修正弱模型边缘节点的局部决策，而不是直接替代弱模型自身的长程规划能力。

4.3 消融实验

为进一步区分云端经验回流机制与强弱协同经验来源的作用，本节在相同弱模型执行器下比较三种设置。*w/o Cloud* 表示在本文执行框架内关闭云端反馈后的纯本地变体，即弱模型仅依赖当前观测与本地历史进行决策，不接收云端总结和反馈；*w/ Cloud* (S1) 表示引入云端异步总结、经验整理与反馈回流后的设置，对应前文场景一；*w/ Cloud* (S2) 表示云端记忆中进一步纳入强模型节点轨迹后的设置，对应前文场景二。

需要注意的是，*w/o Cloud* 用于组件消融，不等同于表 4-2 中的 HiAgent 式本地对照结果。在完整任务表中，LOCAL 与 *w/o Cloud* 含义一致，SCENARIO 1 与 *w/ Cloud* (S1) 含义一致，SCENARIO 2 与 *w/ Cloud* (S2) 含义一致。表中增量均相对于 *w/o Cloud* 计算，其中 SR、PR 和 GA 的增量表示百分点变化，Steps 表示平均执行步数变化，Context 表示相对于纯本地执行设置的归一化上下文消耗变化。

表 4-4 给出了 Blocksworld 上最清晰的组件差异。纯本地执行设置的成功率为 0.00，而两种云端增强设置均带来 20.00 个百分点的成功率增益；任务推进率方面，场景一云端设置提升 25.56 个百分点，场景二云端设置提升 26.67 个百分点。与此同时，两种云端增强设置的平均执行步数分别减少 2.02 步和 2.20 步，说明任务完成和推进效果并非

表 4-4 Blocksworld 任务上的消融实验结果

Model	SR ↑		PR ↑		Steps ↓		Context ↓		GA ↑	
w/o Cloud	0.00		6.67		30.00		100.00%		100.00	
w/ Cloud (S1)	**20.00**	+20.00	32.22	+25.56	27.98	-2.02	106.84%	+6.84%	**100.00**	+0.00
w/ Cloud (S2)	**20.00**	+20.00	**33.33**	+26.67	**27.80**	-2.20	102.68%	+2.68%	**100.00**	+0.00

表 4-5 同一弱模型下的任务级消融实验结果

	SR ↑		PR ↑		Steps ↓		Context ↓		GA ↑	
Blocksworld										
LOCAL	0.00		6.67		30.00		**100.00%**		**100.00**	
SCENARIO 1	**20.00**	+20.00	32.22	+25.56	27.98	-2.02	106.84%	+6.84%	**100.00**	+0.00
SCENARIO 2	**20.00**	+20.00	**33.33**	+26.67	**27.80**	-2.20	102.68%	+2.68%	**100.00**	+0.00
Gripper										
LOCAL	**0.00**		3.50		**30.00**		**100.00%**		**100.00**	
SCENARIO 1	**0.00**	+0.00	2.94	-0.56	**30.00**	+0.00	102.81%	+2.81%	**100.00**	+0.00
SCENARIO 2	**0.00**	+0.00	**6.42**	+2.92	**30.00**	+0.00	100.50%	+0.50%	**100.00**	+0.00
Tyreworld										
LOCAL	10.00		**37.67**		29.20		100.00%		**48.30**	
SCENARIO 1	10.00	-0.00	32.11	-5.56	27.62	-1.58	100.51%	+0.51%	46.14	-2.16
SCENARIO 2	**12.50**	+2.50	32.50	-5.17	**27.25**	-1.95	**92.36%**	-7.64%	41.69	-6.62
Barman										
LOCAL	**5.00**		5.00		**28.75**		**100.00%**		**37.67**	
SCENARIO 1	2.50	-2.50	**6.39**	+1.39	29.35	+0.60	118.62%	+18.62%	27.08	-10.58
SCENARIO 2	**5.00**	+0.00	5.00	+0.00	28.80	+0.05	101.18%	+1.18%	36.44	-1.22
Jericho										
LOCAL	**0.00**		8.91		**29.50**		100.00%		97.00	
SCENARIO 1	**0.00**	+0.00	6.27	-2.64	29.53	+0.03	98.84%	-1.16%	96.50	-0.50
SCENARIO 2	**0.00**	+0.00	**11.62**	+2.72	30.00	+0.50	**95.63%**	-4.37%	**97.33**	+0.33
Overall										
LOCAL	3.00		12.35		29.49		100.00%		**76.59**	
SCENARIO 1	6.50	+3.50	15.99	+3.64	28.90	-0.59	106.64%	+6.64%	73.94	-2.65
SCENARIO 2	**7.50**	+4.50	**17.77**	+5.43	**28.77**	-0.72	**98.40%**	-1.60%	75.09	-1.50

依赖更长的交互过程获得。需要注意的是，在该任务上 Context 相对于纯本地执行设置有所增加，说明云端反馈会引入额外提示内容；但该额外开销伴随成功率和任务推进率的明显提升，体现出云端反馈在规划类任务中的有效性。

表 4-5 中的完整消融结果进一步说明，云端机制的收益不是在所有任务和所有指标上均匀出现，而是与任务结构和经验可迁移性密切相关。总体结果中，场景一相对于本地设置带来 3.50 个百分点的成功率增益和 3.64 个百分点的任务推进率增益，平均执行步数减少 0.59 步；场景二的成功率和任务推进率增益进一步扩大至 4.50 和 5.43 个百分点，平均执行步数减少 0.72 步，并将归一化 Context 降低 1.60%。这与前文场景一和场景二的主实验口径一致：云端经验回流首先改善弱模型的任务推进与完成能力，而强节点经验主要提供补充性增益。

从任务层面看，Blocksworld 对云端反馈最敏感，说明阶段目标清晰、动作前置条件明确的规划任务更容易从轨迹总结和经验回流中获益。Gripper 在场景二中取得 2.92 个百分点的任务推进率增益，但成功率仍未提升，表明强节点经验能够帮助弱模型识别局部状态转移方向，却不一定足以突破最终任务完成所需的全局规划限制。Tyreworld 和 Barman 的结果则更复杂：前者在场景二中成功率提升 2.50 个百分点且 Context 降低 7.64%，但任务推进率和动作落地准确率低于纯本地设置；后者在场景一中任务推进率有所提升，但成功率、执行步数、Context 和 GA 均出现退化。这说明在对象组合、动作约束和目标依赖更复杂的任务中，云端反馈如果不能被弱模型稳定地映射到当前可执行动作，就可能只改善部分中间指标。

Jericho 的结果也体现了这一限制。该任务依赖探索和隐式世界状态建模，场景二带来 2.72 个百分点的任务推进率增益、4.37% 的 Context 降低和 0.33 个百分点的 GA 增益，但成功率仍未提升。这表明强节点经验能够帮助弱模型边缘节点在局部交互上保持更好的推进趋势，却不能完全替代弱模型自身对文本环境的理解和长期记忆能力。综合来看，消融实验支持两个结论：其一，云端经验回流机制是提升弱模型边缘节点任务表现的关键因素；其二，强节点经验能够提供额外帮助，但其效果取决于经验是否能与弱模型当前目标、环境状态和可执行动作保持一致。

4.4 局限性分析

尽管本章实验已经通过两个典型场景和消融实验验证了云边协同经验回流机制对弱模型边缘节点的促进作用，但当前结论仍主要建立在有限任务集合与特定模型组合之上。本文实验所覆盖的 Blocksworld、Gripper、Tyreworld、Barman 和 Jericho 能够反映长时程任务中的多步依赖问题，但对于更复杂的真实边缘环境、更多模型配置以及更强动态性的交互条件，仍需要后续实验进一步检验。

此外，本文当前主要围绕任务完成效果、执行步数、上下文开销和动作落地准确率展开分析，重点说明经验回流机制对弱模型任务能力的影响。对于云端反馈质量、通信时延波动以及不同经验归档策略的细粒度差异，尚未在本章中展开更深入的对比分析。因此，本章实验结果更适合被理解为对方法有效性的验证，而非对所有影响因素的穷尽

性讨论。

4.5 本章小结

本章围绕所提出的云边协同长任务处理方法开展实验验证，给出了实验环境、任务设置、评价指标与实现细节，并通过场景一、场景二和消融实验考察云端经验回流机制对弱模型边缘节点长任务处理能力的影响。

从当前结果看，场景一表明，在不改变边缘模型参数的前提下，云端经验回流机制能够在总体上提升弱模型边缘节点的任务推进效果，并降低上下文消耗和平均执行步数。场景二进一步说明，当云端经验来源包含强模型节点轨迹时，弱模型边缘节点在成功率、任务推进率和动作落地准确率上仍能获得额外提升，但提升幅度相对有限。消融实验进一步表明，云端经验回流机制是弱模型边缘节点获得增益的关键来源，而强节点经验的额外作用受任务结构、经验匹配程度和弱模型自身执行能力共同影响。这表明云端经验汇聚机制能够增强弱模型边缘节点的长时程任务执行能力，同时这种增强仍受到弱模型自身能力与任务结构差异的约束。

5 结论与展望

5.1 研究总结

围绕资源受限边缘环境下大语言模型智能体的长任务处理问题，全文完成了从问题分析、方法设计到实验验证的整体研究。结果表明，以边缘工作记忆控制、云端长期经验整理与反馈回流为核心的云边协同方法，能够在不更新边缘模型参数的前提下提升弱模型边缘节点的长任务处理能力，并在一定程度上兼顾边缘侧的轻量化与低时延要求。其中，场景一验证了云端经验回流机制对弱模型边缘节点的直接增益，场景二则进一步表明，强节点经验经云端整理后能够为弱模型边缘节点带来额外但有限的补充作用。消融实验进一步说明，云端经验回流机制是弱模型边缘节点获得增益的关键来源，而强节点经验的效果受到任务结构、经验匹配程度和弱模型自身执行能力的共同约束。

5.1.1 研究贡献

结合摘要中的主要工作，相关贡献可归纳为四点。

第一，建立了边缘在线执行与云端知识管理相结合的方法架构，明确了边缘侧负责低时延交互与任务推进、云端侧负责阶段轨迹评估、经验归档与知识反馈的职责划分，为资源受限边缘场景中的长任务协同处理提供了整体框架。

第二，形成了子目标驱动的边缘执行机制和分层记忆组织方式，通过当前阶段细粒度保留、历史阶段摘要化维护控制工作记忆规模，降低长任务中的上下文冗余，并保持任务推进的连续性。

第三，构建了云端短期缓存与长期知识库两层知识管理机制，使阶段性执行结果能够转化为可复用的局部反馈与全局指导，并在后续决策中实现经验回流、跨阶段复用与跨节点共享。

第四，实现了云边协同长任务处理方法原型，并补充面向运行过程的可视化监测模块，用于提升云边协同行为的可观测性与实验分析支撑能力；同时，通过 PDDL 规划任务与 Jericho 文本交互任务上的两类实验场景验证了该方法的有效性：场景一验证云端经验回流机制对弱模型边缘节点的直接促进作用，场景二验证强节点经验对弱模型边缘节点的进一步补充作用；并通过消融实验进一步区分云端经验回流机制与强节点经验来源的影响。

5.1.2 研究创新

创新性主要体现在两个方面。

第一，针对相关研究中云边协同偏重计算协同、记忆研究偏重单机组织的不足，云边协同的关注重点被进一步由传统的计算卸载延伸到经验回流驱动的认知协同。由此，云端不再只是额外算力来源，而被界定为阶段经验整理、知识沉淀与跨节点共享的核心支撑节点。这一定位使研究视角区别于仅关注推理协同或单一记忆优化的既有工作。

第二，针对相关研究中工作记忆控制、长期经验复用与反馈回流缺乏统一方法视角的问题，形成了边缘工作记忆控制、云端长期经验沉淀和阶段性反馈回流的一体化设计思路，将工作记忆、长期经验和反馈增强纳入同一闭环。相较于仅依赖本地历史拼接或单一记忆模块的方式，该设计更强调长任务过程中经验的持续形成、回流与复用。

5.2 研究展望

尽管本文对云边协同大语言模型智能体在边缘长任务场景中的应用进行了系统性探索，但仍有若干关键问题亟待在未来工作中进一步深化。

本文现有评测主要基于 PDDL 规划任务与 Jericho 文本交互任务，虽能反映长时程决策与记忆利用特征，但相较真实边缘场景，其环境动态性与感知维度仍显不足。未来工作需扩展任务类型与部署场景，在更高拟真的边缘任务中验证系统表现。

针对云端协同机制，现有设计主要依赖阶段摘要、局部反馈与全局指导作为核心知识单元，但异构任务对经验粒度、反馈时序及知识表示的需求存在差异。后续研究可引入细粒度经验筛选、知识去噪与语义相关性检索机制，探索基于当前子目标、任务状态与节点算力动态自适应选择最优反馈内容的方法，从而提升经验利用率并最小化边缘上下文开销。

除了逻辑层面的知识管理，物理层面的通信开销、反馈时延与任务收益间的权衡关系同样亟待量化分析。针对云边通信受限、网络抖动或云端处理高延迟等极端工况，未来可结合边缘网络实测数据，设计自适应上传频率、异步分级反馈及离线经验缓存等网络感知型协同策略，提升系统在复杂环境下的部署鲁棒性。

在多节点协同维度，跨节点经验共享机制的深度仍有拓展空间。本文场景二结果初步表明，强节点经验经云端整理后能够为弱模型边缘节点带来额外帮助，但这种帮助仍表现为有限增益，并受到弱模型自身能力、任务结构和经验适配性的约束；同时，现阶段的共享形式仍主要依赖云端集中整理的文本态经验。后续可重点突破节点间的任务表征相似度度量、经验置信度评估及个性化知识分发机制，确保共享经验兼具全局通用性与局部适配性。

伴随轻量级边缘大模型基础能力的持续跃升，实现参数高效微调（PEFT）、外部记忆增强与云边反馈机制的深度融合将成为极具潜力的演进方向。相较于本文冻结边缘模型参数的基础设定，探索轻量化参数动态适配与外部知识反馈的联合优化方案，将有望进一步赋予边缘智能体在复杂长任务中的终身学习与持续进化能力。

参考文献

[1] OpenAI. Introducing ChatGPT [J]. OpenAI Blog, 2022.

[2] Team G, Anil R, Borgeaud S, et al. Gemini: a family of highly capable multimodal models [J]. arXiv preprint arXiv:2312.11805, 2023.

[3] 舒文韬，李睿潇，孙天祥，等．大型语言模型: 原理、实现与发展 [J]．计算机研究与发展，2024，61 (2)：351–361．

[4] Li X, Wang S, Zeng S, et al. A survey on LLM-based multi-agent systems: workflow, infrastructure, and challenges [J]. Vicinagearth, 2024, 1 (1): 9.

[5] Wang L, Ma C, Feng X, et al. A survey on large language model based autonomous agents [J]. Frontiers of Computer Science, 2024, 18 (6): 186345.

[6] 奚志恒，陈文翔，郭昕，等．基于大语言模型的智能体: 发展与未来展望 [J]．中国科学: 信息科学，2026，56 (2)：485–486．

[7] Hu M, Zhao P, Xu C, et al. Agentgen: Enhancing planning abilities for large language model based agent via environment and task generation [C]//Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 1. 2025: 496–507.

[8] Schick T, Dwivedi-Yu J, Dessi R, et al. Toolformer: Language models can teach themselves to use tools [J]. Advances in Neural Information Processing Systems, 2023, 36: 68539–68551.

[9] Nguyen D S, Nguyen M T, Nguyen P T, et al. Automated summarization of software documents: an LLM-based multi-agent approach [J]. Automated Software Engineering, 2026, 33 (2): 43.

[10] Raptis E K, Kapoutsis A C, Kosmatopoulos E B. Agentic LLM-based robotic systems for real-world applications: a review on their agenticness and ethics [J]. Frontiers in Robotics and AI, 2025, 12: 1605405.

[11] Zhu L, Huang X, Sang J. A llm-based controllable, scalable, human-involved user simulator framework for conversational recommender systems [C]//Proceedings of the ACM on Web Conference 2025. 2025: 4653–4661.

[12] Park J S, O'Brien J C, Cai C J, et al. Generative Agents: Interactive Simulacra of Human Behavior [C]//Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology. 2023: 1–22.

[13] Wang G, Xie Y, Jiang Y, et al. Voyager: An Open-Ended Embodied Agent with Large Language Models [J]. arXiv preprint arXiv:2305.16291, 2023.

[14] Nakano R, Hilton J, Balaji S, et al. WebGPT: Browser-assisted Question-Answering with Human Feedback [J]. arXiv preprint arXiv:2112.09332, 2021. DOI:10.48550/arXiv.2112.09332.

[15] 白辰甲，许华哲，李学龙．大模型驱动的具身智能: 发展与挑战 [J]．中国科学: 信息科学，2024，54 (9)：2035–2082．

[16] 王文晟，谭宁，黄凯，等．基于大模型的具身智能系统综述 [J]．自动化学报，2025，51 (1)：1–19．DOI:10.16383/j.aas.c240542．

[17] Zhang Z, Dai Q, Bo X, et al. A survey on the memory mechanism of large language model-based agents [J]. ACM Transactions on Information Systems, 2025, 43 (6): 1–47.

[18] Packer C, Wooders S, Lin K, et al. MemGPT: Towards LLMs as Operating Systems [J]. arXiv preprint arXiv:2310.08560, 2023.

[19] Ong K T-i, Kim N, Gwak M, et al. Towards lifelong dialogue agents via timeline-based memory management [C]//Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers). 2025: 8631–8661.

[20] Wang W, Dong L, Cheng H, et al. Augmenting language models with long-term memory [J]. Advances in Neural Information Processing Systems, 2023, 36: 74530–74543.

[21] Yao S, Zhao J, Yu D, et al. ReAct: Synergizing Reasoning and Acting in Language Models [J]. arXiv preprint arXiv:2210.03629, 2022.

[22] Shinn N, Cassano F, Gopinath A, et al. Reflexion: Language agents with verbal reinforcement learning [J]. Advances in Neural Information Processing Systems, 2023, 36: 8634–8652.

[23] Ma C, Zhang J, Zhu Z, et al. Agentboard: An analytical evaluation board of multi-turn llm agents [J]. Advances in Neural Information Processing Systems, 2024, 37: 74325–74362.

[24] Zhai Y, Yang T, Xu K, et al. Enhancing decision-making for llm agents via step-level q-value models [C]//Proceedings of the AAAI Conference on Artificial Intelligence. 2025, 39: 27161–27169.

[25] Hu M, Chen T, Chen Q, et al. Hiagent: Hierarchical working memory management for solving long-horizon agent tasks with large language model [C]//Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). 2025: 32779–32798.

[26] Zheng J, Shi C, Cai X, et al. Lifelong Learning of Large Language Model Based Agents: A Roadmap [J]. 2025. DOI:10.1109/TPAMI.2025.3650546.

[27] Zhang Z, Dai Q, Li R, et al. Learn to Memorize: Optimizing LLM-based Agents with Adaptive Memory Framework [J]. arXiv preprint arXiv:2508.16629, 2025.

[28] Zhou H, Chen Y, Guo S, et al. Memento: Fine-Tuning LLM Agents without Fine-Tuning LLMs [J]. arXiv preprint arXiv:2508.16153, 2025.

[29] Li Y, Qian C, Xia Y, et al. Cross-Task Experiential Learning on LLM-based Multi-Agent Collaboration [J]. arXiv preprint arXiv:2505.23187, 2025.

[30] Xu M, Du H, Niyato D, et al. Unleashing the power of edge-cloud generative AI in mobile networks: A survey of AIGC services [J]. IEEE Communications Surveys & Tutorials, 2024, 26 (2): 1127–1170.

[31] Qu G, Chen Q, Wei W, et al. Mobile Edge Intelligence for Large Language Models: A Contemporary Survey [J]. IEEE Communications Surveys & Tutorials, 2025, 27 (6): 3820–3860. DOI:10.1109/COMST.2025.3527641.

[32] 施巍松，张星洲，王一帆，等．边缘计算: 现状与展望 [J]．计算机研究与发展，2019，56 (1): 69–89．

[33] Yu Z, Wang Z, Li Y, et al. Edge-llm: Enabling efficient large language model adaptation on edge devices via unified compression and adaptive layer voting [C]//Proceedings of the 61st ACM/IEEE Design Automation Conference. 2024: 1–6.

[34] April A. Introducing Apple's On-Device and Server Foundation Models [EB/OL]. (2025).

[35] 王睿，张留洋，高志涌，等．面向边缘智能的大模型研究进展 [J]．计算机研究与发展，2025，62 (10)：2565–2582．

[36] Lin Z, Bi S, Zhang Y-J A. Optimizing AI service placement and resource allocation in mobile edge intelligence systems [J]. IEEE Transactions on Wireless Communications, 2021, 20 (11): 7257–7271.

[37] Luo R, Gu C, He Q, et al. Sim-LLM: Optimizing LLM Inference at the Edge through Inter-Task KV Reuse [J]. Advances in Neural Information Processing Systems, 2026, 38: 89205–89229.

[38] 任姚丹珺，戚正伟，管海兵，等．工业互联网边缘智能发展现状与前景展望 [J]．中国工程科学，2021，23 (2)：104–111．

[39] Liu Z, Zhao C, Iandola F, et al. MobileLLM: Optimizing Sub-billion Parameter Language Models for On-Device Use Cases [C]//Forty-first International Conference on Machine Learning. 2024.

[40] Yao Z, Tang Z, Yang W, et al. Enhancing llm qos through cloud-edge collaboration: A diffusion-based multi-agent reinforcement learning approach [J]. IEEE Transactions on Services Computing, 2025.

[41] 王睿，齐建鹏，陈亮，等．面向边缘智能的协同推理综述 [J]．计算机研究与发展，2023，60 (2)：398–414．

[42] 王睿，王岩，尹朴，等．面向边缘智能的协同训练研究进展 [J]．工程科学学报，2023，45 (8)：1400–1416．DOI:10.13374/j.issn2095-9389.2022.09.26.004．

[43] Zeng A, Liu M, Lu R, et al. Agenttuning: Enabling generalized agent abilities for llms [C]//Findings of the Association for Computational Linguistics: ACL 2024. 2024: 3053–3077.

[44] 袁雷，张子谦，李立和，等．开放环境下的协作多智能体强化学习进展 [J]．中国科学: 信息科学，2025，55 (2)：217–268．

[45] Wegner D M. Transactive memory: A contemporary analysis of the group mind[M]//Theories of group behavior. New York, NY: Springer New York, 1987: 185–208.

[46] Xi Z, Ding Y, Chen W, et al. Agentgym: Evaluating and training large language model-based agents across diverse environments [C]//Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). 2025: 27914–27961.

[47] Gao C, Lan X, Lu Z, et al. S^3: Social-Network Simulation System with Large Language Model-Empowered Agents [J]. arXiv preprint arXiv:2307.14984, 2023.

[48] Borzilov A, Skrynnik A, Panov A. CoSMAC: A Benchmark for Evaluating Communication and Coordination in LLM-Based Agents [C]//LLM-based Multi-Agent Systems: Towards Responsible, Reliable, and Scalable Agentic Systems.

[49] Song H, Goknil A, Jiang X, et al. Developing Multi-Agent LLM Applications Through Continuous Human-LLM Co-Programming [C]//2025 IEEE/ACM 4th International Conference on AI Engineering–Software Engineering for AI (CAIN). IEEE, 2025: 42–47.

[50] Yang Y, Chai H, Shao S, et al. AgentNet: Decentralized Evolutionary Coordination for LLM-Based Multi-Agent Systems [J]. Advances in Neural Information Processing Systems, 2026, 38: 107309–107336.

[51] Xi Z, Chen W, Guo X, et al. The rise and potential of large language model based agents: A survey [J]. Science China Information Sciences, 2025, 68 (2): 121101.

[52] Lindenbauer T, Slinko I, Felder L, et al. The Complexity Trap: Simple Observation Masking Is as Efficient as LLM Summarization for Agent Context Management [J]. arXiv preprint arXiv:2508.21433, 2025.

[53] Ding C, Lu Z, Juefei-Xu F, et al. Towards transmission-friendly and robust CNN models over cloud and device [J]. IEEE Transactions on Mobile Computing, 2022, 22 (10): 6176–6189.

[54] Shao C, Hu X, Lin Y, et al. Division-of-Thoughts: Harnessing Hybrid Language Model Synergy for Efficient On-Device Agents [C]//Proceedings of the ACM on Web Conference 2025. 2025: 1822–1833.

[55] Yao Z, Tang Z, Lou J, et al. Velo: A vector database-assisted cloud-edge collaborative llm qos optimization framework [C]//2024 IEEE International Conference on Web Services (ICWS). IEEE, 2024: 865–876.

[56] Jiang A Q, Sablayrolles A, Mensch A, et al. Mistral 7B [J]. arXiv preprint arXiv:2310.06825, 2023.

[57] OpenAI. GPT-4 Technical Report [J]. arXiv preprint arXiv:2303.08774, 2023.

[58] Vallati M, Chrpa L, Grzes M, et al. The 2014 International Planning Competition: Progress and Trends [J]. AI Magazine, 2015, 36 (3): 90–98.

[59] Hausknecht M J, Ammanabrolu P, Cote M-A, et al. Interactive fiction games: A colossal adventure [C]//Proceedings of the AAAI Conference on Artificial Intelligence. 2020, 34(05): 7903–7910.