基于大语言模型的多结局游戏任务自动化冒烟测试

Jinyu Cai[1], Mingyue Zhang[2], Nianyu Li[3], Jinglue Xu[4],Tianshui Li[5],Jialong Li[1, *], Hitoshi Iba[5],和 Kenji Tei[6]

arXiv:2611.32089v1 [cs.SE] 24 Nov 2026

摘要——**现代游戏任务日益趋向于更高的自由度和多种结局，使其设计与测试变得更加复杂。游戏冒烟测试旨在快速验证任务功能，但现有自动化测试方法在处理包含分支叙事和多个可能结局的任务时存在明显局限。这些挑战导致结局探索不充分、动作序列冗余，以及交互成本高得难以承受。为应对这些挑战，我们提出了 ENDWEAVER，这是一种专为多结局游戏任务设计的自动化冒烟测试框架。ENDWEAVER 利用大语言模型解读任务描述和游戏环境，生成用于指导全局探索的高层方法，并结合生成的原始任务序列进行细粒度规划。通过将这种多层级规划结构与状态转移图的动态构建相结合，ENDWEAVER 确保全面覆盖所有可能的结局，并实现高效测试。我们在四种游戏环境——Mini RPG、Overcooked Plus、Minecraft 和 PySC2——中将 ENDWEAVER 与基线方法进行了比较。在所有环境中，ENDWEAVER 均能稳定且可靠地完成任务，表现显著优于基线方法。在测试成本方面，ENDWEAVER 仅需数千步，而基线方法则需要数百万步，同时仍能确保全面覆盖所有可能的任务结局。消融研究进一步证明了各框架组件的关键作用，验证了我们方法的有效性和鲁棒性。**

索引词—**冒烟测试、大语言模型、角色扮演游戏**

I. 引言

近年来，游戏产业迅速扩张，玩家对更丰富的叙事自由度和更高的内容多样性提出了更高要求 [1]。任务作为吸引玩家参与的核心载体，如今经常采用多结局结构和高度灵活的分支机制 [2]。虽然这些特性增强了玩家的沉浸感，但也给任务设计和验证带来了相当大的复杂度。与此同时，以季节性更新、周期性活动和数值平衡调整为特征的持续运营服务加快了开发周期，一些工作室每天发布多个内部构建版本，并每周更新数百个任务 [3]。游戏复杂度与频繁迭代周期的结合，使测试对于维持开发速度和稳定的玩家体验质量变得愈发重要。

作为任务开发中的内置步骤，冒烟测试 [4]用于验证新实现的内容是否能够正常执行 [5]。与旨在全面检测漏洞的传统测试方法不同，冒烟测试侧重于尽早提供可执行性方面的信号，确保新创建的任务或关卡能够在没有严重问题的情况下进行游玩 [6]。这种测试旨在开发早期验证核心游戏流程的正确性，在高频开发环境中尤为关键[7]。首先，它能为游戏设计师提供快速、明确的反馈，以确认每次迭代的正确性，并避免游戏设计团队与开发团队之间产生误解 [8]。其次，当核心功能出现故障时，它可以防止继续进行全面测试，从而避免测试资源的高昂浪费 [9]。然而，随着多结局任务变得日益复杂且更新愈发频繁，手动冒烟测试在成本、一致性和覆盖率方面面临着重大挑战，这也凸显了自动化的必要性。

游戏中的自动化游戏测试已得到广泛研究出。它最初源于基于模型的方法，旨在通过结构化表示来模拟游戏行为，包括预定义的场景脚本编写 [10],、重放记录的用户交互 [11], ，以及根据 UML状态图 [12] 和 Petri网等描述游戏逻辑 [13]的形式化模型生成测试用例。随后，为了更好地适应现代游戏的复杂性，研究日益转向基于智能体的方法，特别是利用强化学习（RL）智能体与游戏环境交互，以大规模探索状态空间 [14]–[16]的方法。例如，Zheng等人 [17]将深度强化学习与进化算法相结合，在“获胜”（推进游戏进程）和“探索”（覆盖状态空间）之间取得平衡，从而加强针对性漏洞发现。近年来，为了更好地反映人类玩家的多样化偏好，一些研究提出了以角色为中心的方法，让智能体模拟人类游戏风格，使测试更能代表真实的游戏玩法。这些方法包括采用程序化生成的

[1]Jinyu Cai 和 Jialong Li 就职于日本东京的早稻田大学。[2] Mingyue Zhang 就职于中国重庆的西南大学。[3]Nianyu Li 就职于中国北京的中关村实验室。[4]Jinglue Xu 就职于日本东京的 Sakana AI。[5] Tianshui Li 和 Hitoshi Iba 就职于日本东京的东京大学。[6]Kenji Tei 就职于日本东京的东京科学大学。*通讯作者：lijialong @fuji.waseda.jp。已获 IEEE Transactions on Games 接收发表。DOI：10.1109/TG.2026.3720476。© 2026 IEEE。允许个人使用本材料。对于所有其他用途，无论是在当前还是未来的任何媒体中，包括为广告或促销目的重新印刷/再出版本材料、创作新的汇编作品、转售或再分发至服务器或列表，或在其他作品中重复使用本作品中任何受版权保护的组成部分，均须获得 IEEE 的许可。

角色 [18] ，或根据专家示范 [19], [20], 学习玩家风格的方法，使测试智能体能够涵盖更广泛的玩家行为。

然而，上述方法并不适用于冒烟测试，因为冒烟测试侧重于快速验证多结局任务的可执行性，而非穷尽状态覆盖和发现漏洞。用于冒烟测试时，这些方法的实用性受到三个关键问题的限制：（i）现有方法通常需要数百万甚至数十亿次环境交互才能确保收敛；遇到新任务时，通常还需要重新训练或调整参数，导致交互成本高得令人难以承受；（ii）这些方法，尤其是基于强化学习的方法，通常依赖奖励函数进行优化，不仅会受到稀疏奖励分布的影响，难以有效学习，而且往往会在多结局任务中收敛到单一路径，导致探索不足；（iii）这类方法经常生成包含冗余或低效动作的测试序列，难以产出精简且高质量的最优路径。

为解决这些问题，我们提出了一种新颖的自动化冒烟测试框架 ENDWEAVER，其核心思想有三点：（i）利用大语言模型（LLM）的文本理解能力和广博知识，支持对游戏任务进行推理；（ii）采用多层级规划，确保全面探索和高效测试；（iii）结合状态-动作转移图，消除冗余动作。对于（i），ENDWEAVER 利用大语言模型解析任务描述和游戏环境，从而能够对不同任务进行语义推理，并减少过多的环境交互。对于（ii），ENDWEAVER 采用多层级规划方法，将任务分解为高层抽象方法和低层具体原始任务，从而实现全面探索，覆盖多结局任务中的所有可能路径。对于（iii），ENDWEAVER 纳入状态-动作转移图，记录动作序列和状态转换，以消除冗余节点和动作，从而得到最优路径。

本文的主要贡献如下：

- 我们提出了一种新颖的框架 ENDWEAVER，专为多结局游戏任务的冒烟测试而设计。该框架利用大语言模型理解并推理游戏任务，并结合层次化推理与图结构构建，实现对多个结局全面而高效的探索。

- 为应对测试多结局游戏任务所面临的独特挑战，ENDWEAVER 集成了三个关键组成部分：(i) 利用大语言模型解读任务描述并推理多样化的任务结构，从而减少过多的环境交互；(ii) 将方法生成与原子任务序列规划相结合，在广泛的路径探索与精确的任务执行之间取得平衡；以及 (iii) 设计动态状态-动作转移图，以实现高效的路径覆盖、减少冗余并识别关键路径。

- 我们进行了全面的实验评估，结果表明，与传统基线方法相比，我们的框架在效率、路径多样性和计算成本方面表现出更优的性能。此外，我们还进行了全面的消融研究，评估了各个关键组件（即大语言模型、多层级规划和状态-动作转移图）的贡献。

本文的其余部分组织如下：第二节回顾自动化游戏测试的相关工作，以及大语言模型在游戏领域的研究。第三节定义多结局任务冒烟测试的符号和问题。第四节介绍 ENDWEAVER 的技术细节。第五节通过四款游戏对 ENDWEAVER 进行全面评估。最后，第六节总结全文并探讨未来研究的潜在方向。

II. 相关工作

A. 游戏测试

游戏测试在游戏软件的生命周期中发挥着至关重要的作用，旨在确保游戏在发布前及后续更新期间的稳定性、完整性和整体用户体验 [21]。与传统软件应用程序不同，游戏软件需要特别关注游戏玩法机制、用户界面和性能方面 [22]。

随着现代游戏的复杂性不断提升，传统的手动测试方法越来越难以应对数量庞大的可能游戏状态、动作和叙事分支。对此，研究人员和开发者探索了各种自动化和人工智能驱动的测试方法。以下各节将详细介绍这些方法的演进过程：从基础的脚本自动化，到人工智能辅助技术，最终探讨旨在应对复杂动态测试场景的策略。

游戏测试中的自动化。对于相对简单且固定的游戏场景，自动化测试通常通过编写脚本来执行预定义动作 [3], [23]。在这种方法中，测试人员通过设计可能的动作并据此编写控制脚本，来模拟人类玩家的行为。然而，随着游戏环境日益复杂，编写此类定制脚本的成本和低效问题显著增加。为了提高游戏测试的自动化水平，人工智能（AI）技术已应用于多个领域，包括图形用户界面测试、游戏任务脚本的可行性测试以及功能测试 [24]。

人工智能驱动的测试方法。 人工智能驱动的方法旨在减少人工参与并提高测试效率。一种常见的人工智能测试方法是训练智能体模仿人类行为，以完成指定任务 [20], [25], [26]。然而，这些方法高度依赖人工生成的经验数据质量，而且常受泛化问题的限制，因此难以应用于不同的测试场景。为摆脱对人类经验数据的依赖，一些研究提出通过预定义奖励驱动智能体执行特定行为，从而实现测试目标 [27], [28]。不过，奖励函数的设计仍需针对具体场景，这些奖励驱动的测试与

模仿学习有相似之处。它们的效果有限，因为结果高度依赖奖励的设计方式。因此，难以实现全面且灵活的探索，无法满足彻底测试的需求。

复杂环境中的测试策略。除了简单的目标导向型测试方法外，自动化游戏测试的另一个方向是训练智能体探索游戏环境 [29], , 尝试检测特定游戏状态中的错误。Wuji [17] 使用进化算法生成多样化的智能体，从而更容易探索新的状态，并提高发现错误的可能性。其他研究 [30]通过具有不同偏好的智能体探索环境，但这些偏好由人类定义，并且在探索过程中不会发生变化。此外，基于图的 [31], [32]和基于树的方法 [33],[34]也已应用于游戏探索。随着文本格式的任务描述日益普及，文本分类 [35], [36]及其他自然语言处理方法 [37],[38] 也可以应用于游戏测试。此外，在我们此前与网易合作开展的研究中 [39], , 研究人员探索了传统语言模型（例如 FastText、LTSM），通过将任务描述文本与在线规划相结合来加速冒烟测试。然而，与大语言模型不同，这些传统模型的性能有限，不适用于复杂的任务描述和多结局任务场景。

B. 大语言模型在游戏中的应用

由于大语言模型在自然语言理解、上下文推理和推断方面具有卓越能力，因此人们认为它们在改进和简化游戏设计与开发方面极具潜力。现有研究已经将大语言模型整合到众多游戏内应用中，包括作为玩家 [40],担任非玩家角色 [41], 作为助手协助玩家 [42], 提供游戏评论 [43], 在桌面角色扮演游戏中充当游戏主持人 [44],分析玩家数据 [45], 设计关卡难度 [46], 以及最近通过支持玩家直接交互的多模态模型，甚至作为游戏机制运行 [47]。

大语言模型作为游戏中的玩家。 在上述大语言模型于游戏中的应用中，与本研究最相似的是将大语言模型用作游戏中的玩家，即允许大语言模型扮演玩家，在游戏中做出决策并执行动作，以完成特定任务。要让大语言模型“玩”游戏，一个关键挑战在于将游戏状态和动作转换为大语言模型能够高效处理的输入格式。例如，GATO [48]利用多模态数据集进行训练，直接处理视觉输入，从而在雅达利游戏中达到接近人类的性能。然而，这种方法需要大量高质量的游戏视频数据集，在成本和可扩展性方面带来了重大挑战。

作为另一种选择，一些研究提出了间接交互方法：大语言模型不直接与游戏环境交互，而是依赖预定义脚本或解释器等中间层。这种设置尤其应用于 Minecraft 等开放世界游戏。例如，JARVIS-1 [49] 将多模态模型（MLMs）与大语言模型相结合，以感知环境、规划动作，并通过预定义代码层在开放世界环境中执行复杂任务。这使其能够在 Minecraft 游戏环境中高效规划并稳健地执行任务。类似地，VOYAGER [40] 利用 GPT-4 的代码生成能力与 Mineflayer API 交互。通过动态生成代码片段，VOYAGER 能够完成高级指令（例如“寻找最近的树”），并持续扩充其技能库，以提高任务效率。Plan4MC [50] 提出了一种层次化的多任务框架，将强化学习与大语言模型相结合。通过将复杂的开放世界任务分解为易于管理的子技能，Plan4MC 显著提高了样本效率和任务完成率。DECKARD [51]采用了类似的方法，将大语言模型与强化学习相结合，以推测并验证抽象世界模型（AWM），从而提升在 Minecraft 中执行物品制作等任务时的探索能力和样本效率。除了作为游戏中的玩家或助手，大语言模型也被探索用于在预训练阶段为强化学习智能体提供策略指导。例如，Du 等人 [52]提出了一种新颖的框架（ELLM），由大语言模型为强化学习智能体生成语义明确且符合常识的目标。这些目标源自对环境状态的自然语言描述，作为内在奖励，引导智能体将探索行为转向状态空间中与任务相关的区域。与基于新颖性的探索等传统内在动机方法相比，这种由大语言模型引导的方法能鼓励更有目的、且与下游任务更加契合的行为。

基于大语言模型的规划与推理框架。 除游戏特定应用外，近期基于大语言模型的规划研究进展还包括通用推理框架，例如 ReAct [53], ，该框架将推理轨迹与任务特定动作交错结合，以提高可解释性并减少幻觉；以及思维树 [54],, 通过有目的的搜索与回溯探索多条推理路径。尽管这些方法在问答和数学推理等任务中展现出较强的问题解决能力，但它们主要面向推理密集型场景，通过线性或基于树的探索运行，缺乏针对多结局覆盖或消除状态冗余的明确机制。此外，CodeAct [55] 等基于代码的动作框架通过生成可执行的 Python 代码统一动作空间，从而支持灵活的工具组合与动态动作修订。然而，这类方法旨在处理通用应用程序接口交互，缺乏游戏测试工作流所需的领域特定优化，例如结构化路径回放、覆盖率分析以及与质量保证流水线的集成。与上述研究相比，我们的研究在几个关键方面有所不同。首先，我们从开发者视角而非玩家视角处理这一问题，重点在于改进测试流程，而不是模拟玩家行为。其次，除了直接的用户界面交互外，我们还纳入了更广泛的输入，例如游戏元数据和开发侧信息，使框架能够利用更丰富的上下文细节，从而开展更有针对性且更高效的测试。第三，我们的框架高度重视路径多样性与可靠性，以应对

多结局任务的独特需求，并确保冒烟测试能够涵盖分支叙事和游戏玩法可能性的完整复杂性。

III. 问题 формализация

本节首先介绍本文用于描述游戏环境的核心术语，然后基于这些核心术语，给出一个基于马尔可夫决策过程（MDP）的问题定义。

A. 术语与符号

本文研究具有马尔可夫特性的游戏，其中智能体与游戏环境之间的交互可视为 MDP 的一种变体。为便于后续讨论，下面概述用于描述智能代理与游戏环境之间交互模型的术语、概念和符号。

- S 表示状态集合，涵盖游戏中的所有可能状态，例如玩家位置、各种状态信息和周围环境。
- A 表示动作集合，涵盖智能测试代理能够执行的所有动作，例如移动、拾取物品以及与非玩家角色互动。
- T 是状态转移函数，T：$S \times A \to S$，它定义了当系统处于状态 s_t 且智能体在时间步 t执行动作 a_t 时，系统如何转移到新状态 s_{t+1}。
- Q 是任务描述，为智能代理需要完成的任务提供背景和目标。这些描述通常以自然语言或其他结构化形式呈现，用于引导玩家的决策过程。
- E 是任务完成状态集合，表示任务被视为已完成的状态，例如达成某个特定的游戏结局。
- I 是与游戏开发相关的辅助信息集合，例如源代码、关卡设计和上下文数据。这些信息通常以自然语言或代码形式表示，可作为测试代理在环境中进行交互时的指导资源。

在每个时间步 t，智能测试代理根据当前状态 $s_t \in S$ 和任务描述 Q选择一个动作 $a_t \in A$ 。执行该动作后，系统转移到一个新状态 $s_{t+1} = T(s_t, a_t)$。此过程迭代进行，直到智能体到达任务完成状态 $s_t \in E$。智能体的主要目标是探索所有通向任务完成集合 E中某个状态的可能状态-动作路径，确保全面探索并验证任务执行。形式上，这一任务验证过程可表述为一个状态-动作路径搜索问题：智能体从初始状态 $s_0 \in S$出发，根据任务描述选择恰当的动作，并在连续状态间转移，直至到达任务完成状态 $s_t \in E$ 。

B. 问题定义

虽然 MDP 表述提供了状态和动作的基本模型，但复杂任务往往需要一种结构化方法，将抽象目标转化为具体可执行的动作。受分层任务网络（HTN）规划的启发，我们引入一种分层细化过程，不仅逐步细化任务描述 Q，还利用辅助信息 I 指导每一层细化。这确保推导出的任务和动作与任务目标及底层游戏设计保持逻辑一致。

定义 *1*（游戏-智能体交互）：游戏环境定义为一个六元组 $M = \langle S, A, T, Q, E, I \rangle$，其中 S是状态集合， A是动作集合， T 是状态转移函数， Q 是任务描述， E 是任务完成状态集合， I 是与游戏相关的辅助信息集合。智能代理基于分层设计与游戏环境交互。该分层设计自上而下包含三个层级：方法 $\hat{}\pi$、原始任务 $\hat{}\pi$和动作级执行 π。方法定义为 $\hat{}\pi$：$Q \times I \to \mathcal{G}$，其中 $\mathcal{G}$ 是抽象任务目标集合，每个抽象任务目标 $g_h \in \mathcal{G}$ 都可对应从初始状态到完成状态的一条或多条轨迹。原始任务定义为 $\hat{}\pi : \mathcal{G} \to \mathcal{X}$，其中 $\mathcal{X}$ 是模式序列集合。模式 $g_l \in \mathcal{G}_{low}$ 对应于游戏中一系列语义明确的动作， $\mathcal{G}_{low}$ 是模式空间。动作级执行定义为 π : $\mathcal{G}_{low} \times S \to A$。□

通过这种分层细化，智能体无需直接对整个动作空间进行推理。它从高层任务信息入手，逐步将这些信息细化为可执行策略和任务，最终转化为明确完整的游戏动作。这种方法既简化了规划过程，也确保测试代理能够系统地探索和验证复杂的多结局任务，同时保持与底层游戏设计的一致性。

IV. ENDWEAVER

本节首先介绍 ENDWEAVER 的概述，随后详细说明框架的各个核心组件。最后，我们简要介绍大语言模型提示的设计。

A. 概述ENDWEAVER

为应对现代游戏中多路径、多结局任务冒烟测试所带来的挑战，我们提出了分层规划框架 ENDWEAVER，利用大语言模型进行多层级规划。具体而言，我们将任务规划过程分解为四个层级——根任务、方法、原始任务和输入动作，以确保路径探索高效且多样，同时保证任务执行精准。通过进一步整合状态-动作转移图，我们最大限度地提高路径覆盖率，并有效消除冗余路径。

该框架的主要步骤如图 1 所示：1）**方法生成**：在此阶段，目标是将一个复杂的多结局任务拆解为一组

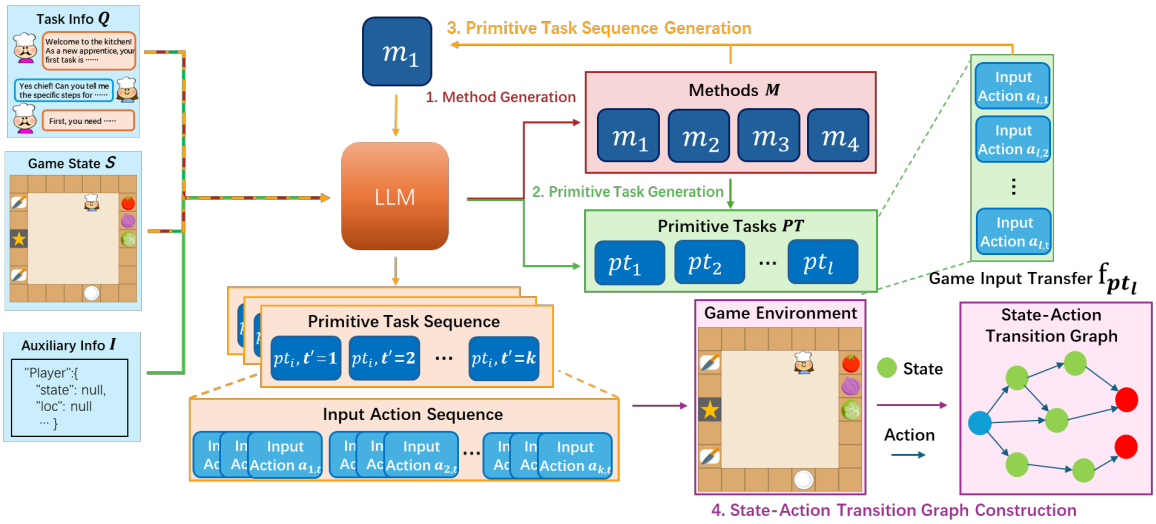


图 1. ENDWEAVER 概览。

高层方法策略，作为完成任务的抽象解决方案。例如，在一个厨房主题游戏场景中，任务是提交一道含番茄的菜肴，框架可以根据任务信息和游戏设计细节推导出多种方法，例如“制作洋葱番茄沙拉”或“制作生菜番茄沙拉”。这样，后续规划过程便可考虑多种完成任务的途径，而非依赖单一固定路径。

2) 原始任务生成与输入动作映射：确定方法后，下一步是将其分解为更小且更易复用的原始任务。继续前面的例子，为了实现“准备洋葱番茄沙拉”这一方法，系统可能会生成“拿起番茄”或“切洋葱”等原始任务。随后，大语言模型生成的函数会将每个原始任务映射到相应的游戏输入动作（例如，按下一系列特定按键），从而使框架能够在游戏环境中执行这些任务。

3)原始任务序列组装：在此步骤中，多个原始任务会被组合成连续序列，以完成所选方法并最终实现游戏目标。例如，为了完成洋葱番茄沙拉，框架可能会生成如下序列：“拿起番茄”，然后“切番茄”；接着“拿起洋葱”，然后“切洋葱”；最后“将食材放在盘子上”。系统可能会生成多个此类序列，以覆盖实现同一目标的不同路径，从而确保全面的测试覆盖率。

4) 状态-动作转移图构建：随着每个原始任务序列的执行，系统会记录相应的游戏状态及其转移，最终构建出一张涵盖所有已探索路径的图。在制作沙拉的场景中，原始任务所需的每个输入动作（例如，拿起食材所需的动作）都会引发一个新的游戏状态。通过追踪这些状态及触发它们的动作，系统构建出状态-动作转移图。这有助于系统地进行路径分析和指标提取，同时消除冗余路线，从而支持对多结局任务中关键分支场景开展针对性测试。

B. 步骤*1*：方法生成

在 ENDWEAVER框架中，我们将多结局任务视为复合任务。这类任务本质上是复杂目标，由多个相互关联的子目标和分支式解决路径构成。为有效应对这些复杂性，我们首先引入方法，将其作为指导如何实现总体任务目标的高层策略。

框架的第一阶段侧重于通过分析任务描述和辅助信息来生成方法。借助大语言模型的自然语言理解和推理能力，我们将复合任务分解为抽象方法。这些方法代表了实现任务完成的多种路径，并为后续的低层规划奠定基础。

方法生成过程可建模为将任务描述和辅助信息映射到一组抽象方法。令 Q 表示任务描述，I 表示辅助信息（例如，游戏设计信息），P 表示为引导大语言模型生成目标而设计的提示。则方法生成过程可表示为：

$$(Q, I, P) \xrightarrow{\text{LLM}} M = \{m_h \mid h = 1, \ldots, n\}$$

其中，$M = \{m_h\}$ 是由大语言模型生成的一组方法。每个 $m_h \in M$ 都代表一种完成任务的可能方法。值得注意的是，m_h 由大语言模型根据其对任务描述和辅助信息的理解生成，主要作为后续规划阶段的指导参考

信息，并反馈输入大语言模型。大语言模型利用大语言模型从输入 Q、 I和 P中推导出 M。 P 是提示，作为一种结构化输入，旨在指导大语言模型生成目标。它提供上下文信息和指令，帮助大语言模型使其输出符合游戏环境的要求以及需要实现的具体目标。

这种抽象通过降低问题的维度简化了解空间，同时保留了高自由度任务所需的灵活性。这些方法在更广泛的任务图中充当引导节点，组织探索过程，并确保系统地覆盖所有可能路径。

C. 步骤*2*：原始任务生成与输入映射

基于这些方法，该框架利用大语言模型生成通用原始任务，这些任务作为规划的基本单元，并被映射为可执行的游戏输入动作。

直接使用大语言模型生成每个具体的游戏动作效率低下。例如，在基于网格的角色扮演游戏中，玩家的许多输入都涉及重复性动作，比如在不同地点之间移动，这些动作不需要高级推理。使用大语言模型生成此类动作会增加计算成本，却没有显著收益。此外，由于大语言模型依赖基于文本的输入和输出，它们更适合生成离散输入，因此在需要连续输入的游戏中的适用性有限。

为应对这些挑战，该框架生成原始任务，而非直接生成游戏输入动作。结合游戏开发信息后，原始任务可以高效地映射为游戏输入动作。这种方法降低了计算开销，并拓展了该框架在离散输入和连续输入游戏中的适用范围。

1) 原始任务生成：原始任务的设计注重可组合性，使得方法能够通过组合和排列这些通用构建块来实现。每个原始任务都被定义为一个结构化元素，以确保清晰性、可复用性，并与后续的原子任务序列规划和输入映射过程兼容。形式上，原始任务生成过程可表示为：

$$(Q, I, M, P) \xrightarrow{\text{LLM}} PT = \{pt_l \mid l = 1, \ldots, m\}$$

其中， $PT = \{pt_l\}$ 是原始任务的集合。每个 $pt_l \in PT$ 都是一个旨在实现可组合性和灵活性的通用任务。这些原始任务充当可复用的构建块，可以组合起来实现各种方法。I 是辅助信息，包括游戏设计介绍以及与指令相关的其他上下文信息。

每个原始任务 $pt_l \in PT$ 都定义为一个三元组：

$$pt_l = \langle \text{Desc}_l, \text{Pre}_l, \text{Effect}_l \rangle$$

其中，Desc_l 是对任务的文本描述，通常由一个动词和一个目标名词组成（例如“把盘子放在桌上”），以确保任务具有基础性和通用性。Pre_l 表示执行前必须满足的前置条件，以确保任务有效（例如“玩家必须有一个盘子”）；Effect_l 表示完成原始任务后的效果，为后续步骤提供参考（例如“盘子在桌上”）。

这种结构化表示确保每个 pt_l 都清晰、可复用，并与后续步骤兼容。此外，尽管Pre_l 和Effect_l 为规划提供了重要指导，但它们充当的是软约束，而非严格要求，从而确保动作规划与执行的灵活性。

2) 输入映射： 生成原始任务后，通过映射函数 f_{pt_l} 将其映射为可执行的游戏输入动作。该映射函数由大语言模型（LLM）根据原始任务 pt_l 和辅助游戏开发信息 I 生成。形式化地说，映射函数可表示为：

$$(pt_l, I, P) \xrightarrow{\text{LLM}} f_{pt_l}$$

对于每个 $pt_l \in PT$，映射函数 f_{pt_l} 将当前游戏状态 s 转换为一系列游戏输入动作：

$$f_{pt_l} : s \to \mathcal{A}_l = [a_1, a_2, \ldots, a_k]$$

其中， s 是当前游戏状态， $\mathcal{A}_l$ 是对应于 pt_l 的游戏输入动作序列。

例如，原始任务“移动到目标位置”可能会使用 A* 或迪ijkstra 等寻路算法，计算从玩家当前位置到目标位置的最优路径。随后，该路径会被分解为具体的游戏输入动作（例如“向前移动”“向右转”），并与其他命令（例如“拾取”“使用”）结合，以生成完整的动作脚本。

这种方法使该框架能够处理多种游戏输入动作类型，同时确保逻辑一致性和执行效率。

D. 步骤 *3*：原始任务序列生成

生成原始任务后，系统利用大语言模型为每种方法生成详细的任务序列。这些序列作为可执行计划，用于完成指定的方法。

序列生成过程可形式化表示为：

$$(s_t, Q, m_h, PT, P) \xrightarrow{\text{LLM}} \text{Sec}_{m_h}$$

其中， s_t 表示当前游戏状态，Sec_{m_h} 表示为方法 m_h生成的原始任务序列。该序列由一组原始任务组成：

$$\text{Sec}_{m_h} = \{pt_{l,1}, pt_{l,2}, \ldots, pt_{l,k}\}$$

这里，$pt_{l,\ t'}$ 表示为完成方法 m_h而选择的第 t' 个原始任务。

为确保生成的动作序列准确无误，我们采用思维链（CoT） [56]推理和反思机制 [57] 作为提示工程技术，以优化大语言模型的输出。

思维链引导模型逐步进行推理，帮助其建立动作之间的逻辑关系，并避免推理跳跃。具体而言，在生成每个pt_l 时t'，系统会提示大语言模型同时生成三个要素：首先是一个原始任务 $pt_{l,\ t'}$，即指明待完成任务的具体目标描述（例如“移动到

厨房”）；其次是状态分析 $s'_{t'}$，详细说明该动作如何影响当前游戏状态，以及由此产生的变化（例如“玩家到达厨房”）；最后是下一步计划 $P_{t'+1}$，根据更新后的状态和持续目标，概述后续动作或策略（例如“打开冰箱并取出生菜”）。

通过思维链，大语言模型逐步推导出任务的各个阶段，确保每项任务都与目标一致。这种结构使大语言模型能够在考虑当前状态的同时，预判每项动作的后果，从而提高序列动作规划的准确性。

此外，为进一步提高原始任务序列的准确性和有效性，我们引入反思机制来完善任务规划。该机制包含三个连续步骤。首先，大语言模型根据给定的任务描述和当前状态生成初始原始任务序列。其次，进入反思阶段，仔细分析初始序列，以识别并指出其中的逻辑不一致、低效之处或错误。最后，在修订阶段，大语言模型根据反思过程中发现的问题修订并优化初始序列，确保最终任务规划更加准确、连贯且高效。

通过将思维链推理与反思机制相结合，系统确保了动作规划的准确性和连贯性。这种方法不仅提升了初始目标序列的质量，还通过迭代不断完善这些序列，从而提高任务完成效率和成功率。最终，该机制确保执行顺序和状态转换符合预期，在复杂场景中最大限度地提高任务执行的准确性。

对于每种方法，都会生成多个原始任务序列，以丰富状态-动作转移图构建；通过利用序列中每个$pt_{l的t'}$ 所关联的映射函数f_{pt_l} Sec^{m_h}，这些序列会被转换为相应的游戏动作输入，以便与游戏环境进行交互。

E. 步骤*4*：状态-动作转移图构建

随着系统执行动作序列，它会动态构建一个状态-动作转移图，用于记录所有已探索的游戏状态及其转移。在该图中，每个节点表示一个唯一的游戏状态 s_i，并编码为固定长度的向量$\mathbf{s_i} \in \mathbb{R}^d$ ，以确保表示的差异性。状态向量$\mathbf{s_i}$ 封装了游戏在任意时刻该状态的所有相关信息，从而确保每个状态都能被唯一识别。

图中的边表示在游戏状态之间进行转移的动作 a_i 。形式上，状态转移可以描述为：

$$s_i \xrightarrow{a_i} s_{i+1}$$

状态-动作转移图在该框架中具有多种潜在用途：

- 路径分析：该图使系统能够识别从初始状态 s_0 到任务完成状态 $s_n \in E$的所有可行路径。具体而言，系统可以追踪通向最终目标状态 $s_n \in E$的不同动作序列 $\{a_1, a_2, \ldots, a_n\}$。通过遍历图并统计从 s_0 到$s_n \in E$ 的所有唯一边序列，可以计算这类路径的总数。

- 冗余消除：随着系统探索游戏空间，可以过滤掉不会导致任务完成或会重新访问先前已探索状态的冗余或无效路径。通过移除循环路径或不利于目标达成的状态，可确保更高效的探索。

- 多结局覆盖：对于存在多个可能结果的任务，例如在游戏中前往不同目的地，状态-动作转移图会突出显示通往各个可能终点的不同路径。例如，玩家可能需要前往森林或山脉，每个目的地都要求采取不同的动作，例如选择特定路线，或沿途收集必要资源。该图通过绘制前往各个目的地所需的独特步骤，展现这些差异，并根据任务要求动态调整。通过呈现不同的选择及其对应路径，该图确保全面探索所有可能结果，从而能够彻底测试和验证多结局场景。

- 增强可解释性：除了抽象的向量表示之外，状态-动作转移图还可以与可视化工具集成，支持从状态向量反向映射到相应的游戏内快照。通过为每个节点配上视觉线索（例如游戏屏幕截图）或描述性的状态信息，开发者可以更直观地识别和理解各个状态及其转移。这种视觉关联有助于开发者更好地理解游戏玩法、发现问题并传达结果，从而提高调试效率。

F. 提示设计

图2展示了本研究中使用的提示示例。我们通过考虑以下关键要素，系统地设计大语言模型提示。

游戏信息和任务背景。提示的主要任务是确保大语言模型获得充足的游戏信息和任务上下文，从而正确理解任务并做出适当决策。游戏信息包括游戏类型、任务目标、可用资源和玩家状态，所有这些信息都有助于大语言模型在开始任务前快速了解游戏环境。在设计提示时，我们区分两种视角：开发者视角和玩家视角。开发者视角侧重于游戏的技术细节，例如任务结构、关卡设计和可能的行动序列；而玩家视角侧重于玩家与游戏环境的实际交互，帮助大语言模型生成符合玩家需求的行动目标和输入映射。这种视角区分确保大语言模型能够生成

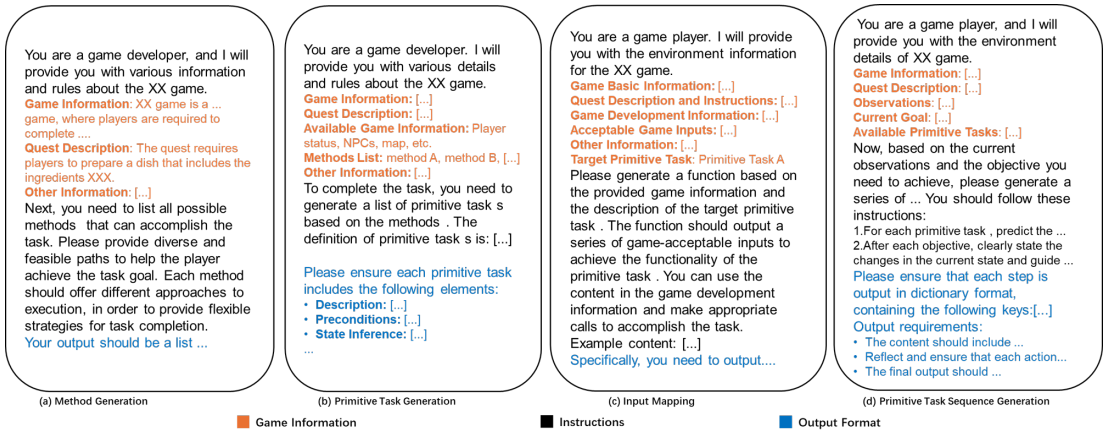


图2. ENDWEAVER 的提示示例

在任务不同阶段均具备上下文准确性的输出。在行动目标生成阶段，开发者视角帮助大语言模型生成有效且可靠的目标，为后续测试提供支持。在行动序列生成阶段，玩家视角提供必要的上下文信息，帮助大语言模型准确规划动作序列并模拟玩家行为。

提示指令。提示中的指令是引导大语言模型生成动作序列的关键。指令会根据任务不同阶段的目标而有所变化。为确保大语言模型准确理解任务并生成高质量步骤，提示必须具备以下几个关键特征：首先，内容应清晰简洁，避免不必要的复杂性，以减少理解错误；其次，应将任务拆解为多个阶段，使大语言模型能够逐步处理，避免信息过载；第三，提示必须与游戏环境紧密契合，确保大语言模型生成的动作符合实际游戏状态、资源和目标。此外，引入思维链和反思机制可显著提升任务生成的质量。思维链通过逐步推理帮助大语言模型验证每个动作的必要性，从而减少任务规划中的潜在错误。反思机制则使大语言模型能够在生成过程中识别并纠正逻辑缺陷，提高所生成序列的整体准确性和连贯性。

输出格式控制。为确保大语言模型的输出结构清晰、可供分析且便于后续处理，我们在提示中对输出格式提出了严格要求。任务输出的每个阶段都应遵循一致的格式，尤其是在生成原始任务和动作序列时，输出应采用字典结构。字典必须包含“行动描述”“状态分析”和“下一步规划”等关键信息，明确说明每项操作的目的、前置条件和预期结果，以便后续分析与执行。这种格式控制不仅确保了任务执行的逻辑连贯性和可追溯性，还能通过结构化输出数据有效地进行任务分析与测试。

V. 评估

本评估旨在回答以下研究问题（RQ）：

- 研究问题 1：与基线方法相比，ENDWEAVER 完成任务的稳定性如何？其成功的行动计划有多简洁？
- 研究问题 2：ENDWEAVER 能否有效地发现并覆盖所有指定的任务结局？
- 研究问题 3：在测试过程中所需的环境交互总步数方面，ENDWEAVER 与基线方法相比表现如何？

A. 实验设置

评估指标。针对上述研究问题，我们采用以下定量指标。对于研究问题 1，我们采用两个关键指标。第一个是完成率，用于衡量成功完成任务的测试运行所占比例。第二个是完成任务所需的平均步数，仅根据成功运行计算，反映实现任务完成所需行动路径的长度。对于研究问题 2，我们统计所有成功运行中每种可能结局（结局 A、B 或 C，难度依次递增）达成的次数，并计算相应比例。对于研究问题 3，我们评估整个测试过程中产生的环境交互总步数，包括训练阶段和测试阶段。

基线方法。我们选取了四种具有代表性的基线方法。所有方法均在相同的底层环境动态和动作语义下运行，同时采用适合各自方法的观测表示：

(i) 随机算法 [39], [58]：从可用动作空间中随机采样动作，直至任务完成或达到最大步数限制，作为衡量无信息探索难度的基准。Mini RPG 的最大步数限制为 5,000 步，Overcooked Plus 为 10,000 步，Minecraft 为 2,000 步，PySC2 为 200 步。每次迭代中，每个任务采样 30 次。

（ii）遗传算法（GA） [59]：一种经典的基于搜索的方法，通过演化固定长度的动作序列来优化

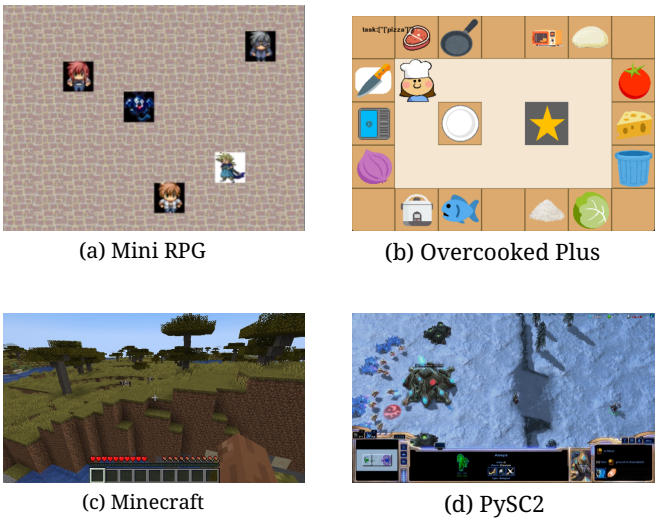
任务完成。Mini RPG 的序列长度为 100，Overcooked Plus 为 1,000，Minecraft 为 50，PySC2 为 200。种群规模为 100；Mini RPG/Overcooked Plus/Minecraft 的最大代数为 150，PySC2 为 50；交叉算子采用 Xovdp，概率为 0.7；变异算子采用 Mutbga，概率为 0.01，收缩参数为 0.5，梯度参数为 20。当连续 10 代未观察到适应度改进时，提前停止机制将终止演化。每次迭代中，每个任务/任务独立训练 10 次，并保留最终种群中排名前 3 的个体用于评估。

(iii) 近端策略优化（PPO） [60]：一种广泛使用的策略梯度强化学习算法，通过优化策略网络来最大化期望回报。我们使用以向量化游戏状态为输入的多层感知机，并设计特定于环境的奖励函数。训练最多使用 500,000 个时间步；如果连续五次评估均未观察到改进，则提前停止。超参数如下：学习率 3×10^{-3}、滚动步数 $n=2048$、批量大小 64、折扣因子 $\gamma=0.99$、GAE 参数 $\lambda=0.95$、裁剪范围 0.1、价值函数系数 0.5、熵系数 0.01、优化轮数 10，以及梯度裁剪阈值 0.5。在一次迭代中，每个任务/任务均使用 10 个独立初始化的模型进行训练，每个训练好的模型均通过 3 次滚动运行进行评估。奖励设置如下：Mini RPG：步进惩罚 -0.1、子任务完成奖励 $+100$；Overcooked Plus：步进惩罚 -0.1、食材处理奖励 $+10$、任务完成奖励 $+1000$、错误提交惩罚-20；Minecraft：步进惩罚 -0.1、获得必需物品奖励 $+10$、任务完成奖励 $+1000$；PySC2：采用混合奖励，包括终局结果奖励，以及基于任务进度、敌方关键目标减少、战场推进和友方单位保存的稠密塑形反馈，遵循以往 PySC2 奖励设计实践 [61]。推理阶段使用随机动作采样，以鼓励探索。

(iv) ELLM-PPO [52]：一种增强型 PPO 方法，引入了大语言模型辅助探索。GPT-4o 根据当前状态描述和任务规范生成自然语言子目标。内在奖励根据状态转移描述与生成的子目标之间的语义相似度计算得出。智能体先使用这一增强奖励进行预训练，然后使用标准 PPO 进行微调。为确保公平，所有强化学习训练超参数（学习率、批量大小、网络架构等）均与 PPO 基线保持一致，唯一的区别是在预训练阶段加入了由大语言模型生成的内在奖励。每次迭代使用相同的 10 个模型、3 次 rollout 协议。

(v) ENDWEAVER（我们的方法）：使用 GPT-4o 作为大语言模型规划器，并结合多层级规划结构和状态—动作转移图来完成任务。对于每种方法，生成 30 个动作序列，并记录在状态—动作转移图中。随后通过合并相同节点并删除冗余路径来处理该图，之后随机抽取 10 条有效的任务完成轨迹作为最终输出。

所有实验均使用随机种子42。为估计统计不确定性，完整设置的实验重复进行30次，所有报告值均以使用t分布构建的均值 ± 95%置信区间表示。这30次重复仅用于置信区间估计，不计入总步数。



(a) Mini RPG (b) Overcooked Plus (c) Minecraft (d) PySC2

图 3. 四个评估游戏环境的代表性截图。

为了全面评估我们的方法，我们在四种不同的游戏环境中开展实验：Mini RPG、Overcooked Plus、Minecraft，以及一个基于 PySC2 的即时战略游戏场景。这些环境涵盖了多样的游戏类型、任务结构和难度特征。

*Mini RPG*游戏。Mini RPG是一款简化的线性剧情角色扮演游戏 [39], ，旨在评估自然语言丰富场景中的任务理解与规划能力。它在保留传统角色扮演游戏核心逻辑的同时，简化了地图和交互，以支持快速生成任务并确保可复现性。该环境包含一张固定的网格地图，其中有非玩家角色、怪物、玩家角色和交互式地形。玩家可以与非玩家角色交互、收集或提交物品、参与战斗并触发脚本事件。游戏状态完全可观测，并以结构化字典表示，其中包括地图布局、角色属性、非玩家角色和活跃任务。该环境设有 9 个任务，每个任务都有多条难度各异的完成路线。每条路线都由一系列以自然语言描述的子任务组成，要求玩家执行不同的动作序列。作为一种简单基础的环境，Mini RPG游戏旨在为该方法提供入门级测试场景，验证其能否在低复杂度任务中稳定实现目标，并为后续在更复杂环境中的评估建立性能基线。

Overcooked Plus。Overcooked Plus [62] 是商业烹饪游戏 Overcooked [63], 的开源增强难度版本，旨在评估操作约束下的规划能力和执行敏感性。玩家通过切菜、烹饪、装盘和上菜等动作来处理并组合食材，制作菜肴。环境施行严格规则——例如，每个工作台一次只能放置一件物品，放错位置的物品必须明确丢弃。

该环境完全可观测，并采用结构化状态表示来追踪所有相关对象的位置和状态信息。任务集包含 7 个烹饪任务，每个任务都有多种有效的完成策略，难度逐步提升。难度较高的任务通常需要更复杂的组合和更精确的动作序列。该环境强调严格规则下的序贯规划，要求智能体保持较高的执行准确性，同时快速响应状态变化并纠正错误。

Minecraft。Minecraft [64] 是一种大规模沙盒环境，因其开放式结构和丰富的组合交互动态而广泛应用于人工智能研究。为支持一致的评估，我们在 ENDWEAVER框架中采用高级动作抽象，所有方法均发出离散的参数化命令，例如“挖掘”“制作”或“移动”。这些命令会在内部映射为低级原子操作。该环境能够完全观测智能体的位置、库存状态、资源位置和工具状态。为确保不同方法之间具有可比性，我们将动作空间离散化为一组固定的参数化高级命令。我们定义了 11 个受 Minecraft 成就系统启发的任务，涵盖资源采集、物品制作和战斗。每个任务都支持多种复杂程度各异的有效解决方案，以鼓励长时域规划和自适应策略。该环境的难点主要源于漫长的依赖链和庞大的状态空间。

PySC2。我们进一步引入一个基于 PySC2 的评估场景 [61], [65] ，以评估方法在更复杂的即时战略游戏环境中的可扩展性。该场景在固定的 Protoss 对阵 Zerg 配置下包含五项任务。与现有环境相比，此设置的决策复杂度更高，且要求智能体根据部分可观测的游戏内信息持续做出实时决策，因此对方法的在线决策能力提出了更高要求。

在观测和动作协议方面，PySC2 的观测仅限于当前游戏窗口中可见的信息，包括进度信号、友方单位状态、可见的敌方信息以及空间位置特征。PPO 和 ELLM-PPO 使用归一化的向量化观测，而 ENDWEAVER 接收经过语义结构化的自然语言观测。PySC2 的动作处理采用与 Minecraft 相同的统一高级动作抽象策略，以确保不同方法之间可以直接比较。实际使用的代表性动作包括空操作、移动至指定位置、攻击目标和技能施放指令，其参数分别以屏幕/小地图坐标、目标单位ID和轻量级控制标志的形式指定。

这五项任务都是具有不同战术设置的战斗场景。对于每项任务，我们手动设定三个难度递增的结局（结局 A/B/C）。结局设计因任务而异，相应的达成条件也各不相同，包括战斗胜利（摧毁指定的敌方单位/建筑）、防御胜利（在规定时长内保护指定的友方单位/建筑）、经济胜利（达到目标资源阈值），以及在预先设定的特殊条件下达成平局。我们指出，这种明确设定多个结局并非 PySC2 原始任务设计的一部分。然而，由于能够同时提供多样化游戏玩法设置和明确多结局结构的开放且可供实验使用的基准测试极为有限，我们在现有任务中构建了这些结局，以便进行受控评估。尽管这种调整并不能完全复现原始游戏设计，但在实际游戏设计中，类似分支式的结果结构十分常见。

这五项任务在敌人组成、可用的友方单位组合、技能配置、地图空间结构和时间压力方面存在系统性差异。这些变化形成了多样的战术情境，要求智能体持续解读部分可观测状态，并在实时约束下准确地逐步做出决策。

B. 实验结果

表 I 汇总了所有四个基准环境中的结果。在任务完成率方面，ENDWEAVER 在所有环境中始终是表现最强的方法。在 PySC2 中，其完成率达到 81.2%，优于 ELLM- PPO（78.6%），并明显高于 PPO（64.6%）、遗传算法（47.3%）和随机（38.9%）。这表明，即使环境变得更加动态，且对执行精度的要求更高，ENDWEAVER 中的语义规划和基于图的轨迹整合仍然有效，不过在这种反应性更强的设置中，其优势明显缩小。

在平均完成步数方面，ENDWEAVER 也展现出明显优势。在 Mini RPG 中，它仅需平均 28.3 步即可完成任务，显著少于所有基线方法。在 Overcooked Plus 和 Minecraft 中，尽管近端策略优化的平均步数更少（分别为 43.8 步和 6.9 步），但仔细查看结局分布会发现，近端策略优化和 ELLM-PPO 都明显偏向最简单的结果——结局 A。例如，在 Minecraft 中，近端策略优化在 89.9% 的成功运行中达成结局 A，而最难的结果结局 C 仅占 2.7%。由于我们的评估将结局难度定义为所需动作序列长度，因此平均步数较少并不一定意味着任务规划更高效或更有效。相比之下，ENDWEAVER 在保持平均步数相对较少的同时，使三种结局的分布更加均衡。在 PySC2 中，ENDWEAVER 仍实现了最短的成功轨迹（34.1 步），显著少于 ELLM-PPO（54.8 步）、遗传算法（66.2 步）、近端策略优化（81.4 步）和随机（89.6 步）。这表明，即使在即时战略场景中，该框架仍能稳定地执行相对简洁的成功规划。

从结局覆盖率来看，在更具挑战性的环境中，各方法之间的差异进一步扩大。虽然大多数方法都能在相对简单的 Mini RPG游戏中有效达成全部三个结局，但在 Overcooked Plus 和 Minecraft 中，性能差异显著。随机方法无法达成任何结局 C，而基于近端策略优化的方法则明显偏向简单路径。遗传算法展现出一定的能力，能够达成高难度结果（例如，在 Minecraft 中达成结局 C 的比例为 12.5%）。相比之下，ENDWEAVER 达到了 30.4%

表 I 四种游戏环境下任务完成与结局多样性的性能指标

Game	方法	完成率(%)	平均步数	结局 A(%)	结局 B(%)	结局 C(%)	总步数
Mini RPG	随机	25.3 ± 3.8	1463.9 ± 155.2	66.7 ± 5.1	24.4 ± 4.5	8.9 ± 2.3	1,001,046
	GA	54.4 ± 4.5	81.0 ± 7.8	58.5 ± 5.2	24.5 ± 4.1	17.0 ± 3.3	124,086,117
	PPO	62.5 ± 4.8	47.2 ± 3.5	74.2 ± 4.1	18.4 ± 3.2	7.4 ± 2.6	10,838,016
	ELLM-PPO	74.6 ± 3.9	53.2 ± 4.9	65.0 ± 4.4	22.1 ± 3.6	13.0 ± 2.9	9,920,000
	ENDWEAVER（我们的方法）	100.0 ± 0.0	28.3 ± 1.6	40.6 ± 3.3	30.7 ± 3.1	28.7 ± 2.8	6,750
Overcooked Plus	随机	49.2 ± 5.1	5686.2 ± 481.6	70.6 ± 5.8	29.4 ± 5.8	0.0 ± 0.0	688,432
	GA	68.5 ± 4.6	753.1 ± 68.3	47.1 ± 5.5	42.9 ± 5.6	10.0 ± 3.1	17,854,231
	PPO	56.2 ± 5.3	43.8 ± 4.1	88.8 ± 2.8	9.3 ± 2.3	1.9 ± 0.9	2,905,484
	ELLM-PPO	78.4 ± 4.1	79.3 ± 7.2	56.5 ± 5.1	33.6 ± 4.4	9.9 ± 2.8	3,510,323
	ENDWEAVER（我们的方法）	100.0 ± 0.0	66.0 ± 3.5	23.2 ± 3.6	46.4 ± 4.3	30.4 ± 3.0	15,036
Minecraft	随机	32.1 ± 4.4	462.4 ± 91.5	74.7 ± 6.3	25.3 ± 6.3	0.0 ± 0.0	431,456
	GA	55.6 ± 5.2	15.3 ± 2.4	62.5 ± 6.1	25.0 ± 4.8	12.5 ± 3.3	5,262,322
	PPO	43.7 ± 5.5	6.9 ± 0.9	89.9 ± 3.1	7.4 ± 2.4	2.7 ± 1.2	335,163
	ELLM-PPO	68.8 ± 4.4	8.6 ± 1.2	69.4 ± 5.5	19.0 ± 4.1	11.6 ± 3.2	293,112
	ENDWEAVER（我们的方法）	100.0 ± 0.0	7.3 ± 0.6	37.0 ± 4.5	34.3 ± 4.2	28.7 ± 3.8	1,413
PySC2	随机	38.9 ± 4.0	89.6 ± 8.5	57.8 ± 5.1	38.0 ± 3.9	4.2 ± 1.5	18,320
	GA	47.3 ± 4.2	66.2 ± 5.7	56.2 ± 4.7	38.1 ± 3.8	5.7 ± 1.8	12,186,540
	PPO	64.6 ± 4.9	81.4 ± 6.2	52.6 ± 4.6	40.1 ± 3.8	7.3 ± 2.0	2,356,904
	ELLM-PPO	78.6 ± 3.9	54.8 ± 4.5	46.8 ± 4.2	42.9 ± 3.6	10.3 ± 2.1	2,401,772
	ENDWEAVER（我们的方法）	81.2 ± 3.5	34.1 ± 2.0	48.7 ± 4.1	44.9 ± 3.3	6.4 ± 2.0	25,406

注： 结局难度定义为到达该结局所需的最少动作数。序列越长，表示难度越高。（PySC2 除外）

的 Overcooked Plus 结局 C 覆盖率，并且在 Minecraft 中实现了近乎均匀的结局分布（结局 A、B 和 C 的比例分别为 37.0%、34.3% 和 28.7%）。在 PySC2 中，情况则更加微妙：ENDWEAVER 仍将覆盖范围扩展到了简单结局之外，并在中等复杂度分支上取得最高比例（结局 B：44.9%），但其结局 C 比例（6.4%）仍低于近端策略优化（7.3%）和 ELLM-PPO（10.3%）。这表明，在即时战略游戏设置中，最难的情况较少依赖长程依赖链推理，而更多取决于快速变化状态下的反应式执行。由于 ENDWEAVER 并非在线规划架构，其离线长时域推理在这种设置下更容易受到影响；而基于近端策略优化的方法则可能偶尔在成功的运行过程中受益于在线适应。

在测试成本方面，与基于训练的基线方法相比，ENDWEAVER 所需的总环境交互步数大幅减少。在 Mini RPG 中仅需 6,750 步，在 Overcooked Plus 中需 15,036 步，在 Minecraft 中需 1,413 步，在 PySC2 中需 25,406 步。相比之下，GA 在 Mini RPG 中需要超过 1.24 亿步，PPO 在该环境中消耗超过 1,000 万步；而在 PySC2 中，Random、GA、PPO 和 ELLM-PPO 分别需要 18,320、12,186,540、2,356,904 和 2,401,772 步。Random 在 PySC2 中的成本低于 ENDWEAVER，这是因为它不进行训练；但较低的交互成本伴随着显著较弱的完成可靠性和困难结局覆盖率。GA、PPO 和 ELLM-PPO 所需的交互步数都明显多于 ENDWEAVER。总体而言，结果表明，与具有竞争力的搜索方法和强化学习基线相比，ENDWEAVER 在保持强大多结局探索能力的同时，实现了显著更优的性能–成本权衡。但与此同时，交互效率并不完全等同于总运行成本，因为 ENDWEAVER 还依赖大语言模型推理。在这种情况下，我们将总步数理解为交互成本，而不是完整端到端的计算成本；关于推理开销的更详细讨论见局限性部分。

C. 消融研究

为系统评估 ENDWEAVER 中各核心组件的贡献，我们在 Overcooked Plus 中开展了消融研究。具体而言，我们考察了三个关键模块的作用：(i) 大语言模型；(ii) 状态—动作转移图（图）；以及 (iii) 多层级规划（MLP）。

大语言模型消融。为评估不同大语言模型的影响，我们保持完整框架不变（“完整设置”，同时保留图和 MLP），并使用 GPT-4o、GPT-4o-mini、Qwen2-72B、DeepSeek-V2 和 Claude-3.5-Haiku 等不同大语言模型进行比较（其他实验设置与第五节 A 部分保持一致）。结果呈现出明显的性能分层：第一梯队模型（如 GPT-4o、DeepSeek-V2、通义千问）不仅能够稳定完成所有任务，还能发现所有结局，包括最复杂的结局 C；第二梯队模型（如 GPT-4o-mini、Claude-3.5-Haiku）在该框架的引导下仍可达到 100% 的完成率，但完全无法到达结局 C。这表明，尽管 ENDWEAVER 的稳健性确保了较高的性能下限——即使较弱的大语言模型也能完成基本任务——但应对最具挑战性的推理问题并探索最深层结局的能力，则完全取决于底层大语言模型的推理能力。

图消融实验（无图）。在“无图”设置下，系统直接从大语言模型的输出中采样动作序列，而不构建状态-动作转移图。如

表 II Overcooked Plus 不同设置下的性能比较

设置	方法	完成率(%)	平均步数	结局 A(%)	结局 B(%)	结局 C(%)	总步数
完整（我们的方法）	DeepSeek-V2	100.0 ± 0.0	69.2 ± 4.0	20.8 ± 3.9	52.0 ± 5.2	27.2 ± 4.0	16,184
	Qwen	100.0 ± 0.0	77.7 ± 4.2	25.3 ± 4.1	41.8 ± 5.1	32.9 ± 4.5	15,428
	GPT-4o	100.0 ± 0.0	66.0 ± 4.2	23.2 ± 4.4	46.4 ± 5.5	30.4 ± 4.7	15,036
	GPT-4o-mini	100.0 ± 0.0	36.4 ± 2.1	42.0 ± 10.0	58.0 ± 10.0	0.0 ± 0.0	14,951
	Claude-3.5-Haiku	100.0 ± 0.0	33.5 ± 1.4	57.0 ± 10.0	43.0 ± 10.0	0.0 ± 0.0	15,363
无图	DeepSeek-V2	26.1 ± 4.3	79.8 ± 4.3	64.3 ± 9.7	20.7 ± 8.6	15.6 ± 8.5	16,149
	Qwen	28.1 ± 5.6	81.2 ± 2.9	45.8 ± 10.8	50.6 ± 9.1	3.6 ± 3.5	15,182
	GPT-4o	46.1 ± 5.2	82.9 ± 2.6	25.7 ± 9.5	59.6 ± 9.4	16.3 ± 7.6	14,823
	GPT-4o-mini	5.6 ± 2.3	69.8 ± 4.1	90.7 ± 12.2	10.6 ± 13.8	0.0 ± 0.0	14,781
	Claude-3.5-Haiku	27.5 ± 5.7	69.0 ± 2.3	32.4 ± 9.6	67.2 ± 10.4	0.0 ± 0.0	15,216
不含 MLP	DeepSeek-V2	100.0 ± 0.0	26.0 ± 1.0	100.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	6,128
	Qwen	100.0 ± 0.0	26.6 ± 1.2	100.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	5,953
	GPT-4o	100.0 ± 0.0	25.8 ± 0.6	100.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	6,761
	GPT-4o-mini	100.0 ± 0.0	25.3 ± 0.4	100.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	5,888
	Claude-3.5-Haiku	100.0 ± 0.0	26.4 ± 1.4	100.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	6,544

表 II 所示，移除图会导致性能显著下降：即使是表现最强的模型 GPT-4o，其完成率也从 100.0% 降至 46.1%，凸显了图在确保任务执行稳定且一致方面发挥的关键作用。没有图的支持，复杂结局的覆盖情况也明显恶化，最困难的结局 C 的完成率急剧下降——某些模型甚至降至零。这是因为状态-动作转移图能够合并和复用共享子路径，从而生成更丰富、更多样的可行轨迹集合，尤其有利于到达更深层或概率较低的结局。此外，没有图时，完成任务所需的平均步数也显著增加。由于缺乏关于已探索状态和转移的全局信息，系统往往会生成更长或冗余的序列。相比之下，图支持路径剪枝和复用，有助于生成更短、更高效的计划。这些发现表明，转移图不仅对于提升执行稳定性至关重要，也能支持高效的轨迹构建，并推动对解空间进行更深入的探索。

多层级规划消融实验（不含 *MLP*）。在“不含 MLP”的设置下，大语言模型仅依据任务描述和游戏指令生成完整的动作序列，随后将其记录到状态-动作转移图中，不进行层次化目标分解，也不提供基于提示的完成偏好引导。结果显示，五个大语言模型在不使用 MLP 的情况下仍保持 100.0% 的完成率；然而，结局多样性完全消失，所有运行结果都收敛到最简单的结局 A（B 和 C 的覆盖率：0.0%）。平均步数也显著下降至约 25–26 步。例如，GPT-4o 在“不含 MLP”设置下的平均步数 / 总步数为 25.8 / 6,761，而在完整设置中为 66.0 / 15,036，结局多样性则从 $A/B/C = 23.2/46.4/30.4$ 变为 $A = 100.0$、$B = C = 0$。DeepSeek-V2 和通义千问也呈现出相同趋势。需要强调的是，平均步数的减少并不表示效率得到提升；相反，这反映了难度分布的转移以及一种指标偏差其原因在于系统只探索最简单的结局。这一现象与第五节 B 中讨论的近端策略优化结果类似。因此，多层级规划结构通过层次化目标分解和路径引导，有效地促使系统进入更长、更复杂的依赖链，从而解锁中等/高难度分支（B/C），并提升多结局覆盖。

D. 讨论

在四个基准环境中，ENDWEAVER 在完成可靠性、结局覆盖率和交互效率方面均展现出持续强劲的性能。在表 I 中，ENDWEAVER 在全部四个环境中都取得了最高的完成率，达成了所有指定的任务结局，并且所使用的环境交互步数比具有竞争力的搜索方法和强化学习基线少了几个数量级。与此同时，PySC2 的结果表明，当主要挑战从长时程依赖推理转向实时反应式控制时，这一优势会明显缩小。

消融研究进一步凸显了各个核心组件的贡献。大语言模型的推理能力决定了系统性能的上限：更强的模型能够持续发现更复杂的结局，而规模较小的模型虽然能够在该框架的引导下保持较高的完成率，却无法到达高难度分支。状态—动作转移图对于执行稳定性和深度探索都至关重要；移除该组件会导致完成率和结果多样性均大幅下降，因为智能体失去了复用子路径、剪枝冗余探索以及全局协调轨迹的能力。多层级规划结构对于多结局覆盖同样至关重要，因为在没有 MLP 的情况下，所有运行都会坍缩到最简单的结局。

相比之下，传统的强化学习方法（例如近端策略优化）和基于搜索的方法（例如遗传算法）从根本上仍受其“搜索式”策略范式的限制，性能在很大程度上依赖于奖励函数的设计。这些方法受到奖励信号稀疏性和外部性的影响

——尤其是近端策略优化，往往会陷入局部最优。在 Overcooked Plus 中，近端策略优化通常会收敛到最简单的结局。我们推测，一个关键瓶颈在于其奖励函数设计的局限性。这一问题在长时域、高复杂度的任务中尤为突出：对于到达复杂结局至关重要的中间动作（例如回溯）可能无法带来即时奖励，从而使策略网络难以学习这些动作。因此，近端策略优化难以探索完整的状态空间，进而难以达成更复杂的结局。要构建如此细粒度的奖励结构，需要深厚的领域知识以及大量试错，因此在复杂游戏场景中实现起来困难且成本高昂。值得注意的是，表 I 所示结果并未计入调试奖励函数的成本；若将其纳入考量，总开销还会进一步增加。借助大语言模型辅助的内在奖励，ELLM-PPO 在一定程度上缓解了局部最优问题，但从根本上仍受强化学习探索范式局限性的约束。由于遗传算法具有全局搜索特性，其多样性略优于近端策略优化，但代价是样本复杂度极高且计算开销高昂。

为解决这些局限，ENDWEAVER 采取了截然不同的视角：立足于语义层规划，而非奖励驱动搜索。通过在执行前明确推理最终目标及其逻辑依赖关系，ENDWEAVER 能够从全局视角生成有目的且多样化的动作轨迹，有效避免困扰基于强化学习的方法的局部最优。此外，这种架构具有很强的可扩展性：适配新的环境或任务时，只需指定游戏机制、应用程序接口，以及原始任务与游戏操作之间的映射关系，从而构建一条知识工程化流程；无需重新训练成本高昂的奖励函数或策略网络。这种设计有望在保持长程复杂任务良好性能的同时，降低迁移和部署成本，不过 PySC2 的结果也揭示了它在高度依赖反应速度的场景中的明显局限。已有研究表明，基于大语言模型的在线规划具有潜力；因此，ENDWEAVER 的一个自然扩展方向，是在高度依赖反应速度的场景中用在线生成取代离线原始任务生成，从而改善实时适应能力和任务完成效果。另一个值得探索的方向是将 ENDWEAVER 与专用的局部规划或控制算法相结合，让大语言模型专注于叙事层推理和高层依赖管理，而由低层规划器处理细粒度的操作决策。这种混合设计或许能更好地发挥语义规划与反应式控制在高度动态环境中的互补优势。

除了作为初始冒烟测试的有效工具之外，我们的框架还有望在整个游戏测试生命周期中持续发挥价值。构建的状态—动作转移图汇集了有关任务结构和玩家—环境交互的丰富知识，可在各种测试活动中重复利用并加以扩展。例如，在回归测试中，可以快速重放此前发现的关键路径和结局，以验证游戏更新后关键功能是否仍然完好，同时将成本降低几个数量级，相较于完整重新探索，成本可降低几个数量级。在测试覆盖率分析中，该图能够系统地检查已探索和未探索的分支，为未来测试的优先级排序提供指导。

这些特性也表明，该框架可以进行实用的人机交互扩展。一种潜在的界面可以提供已探索状态和结局的基于图的可视化、可编辑任务计划，以及将已验证轨迹一键导出为工程产物的功能。这类可视化还可以通过为设计师提供策略空间的透明概览，帮助识别非预期的捷径、死胡同或逻辑缺陷，从而辅助任务设计验证。虽然我们将正式的可用性评估留待未来工作，但当前的结构化输出和转移图已经使此类扩展在技术上可行，并且与以测试人员为中心的工作流程高度契合。

E. 局限性

尽管实验结果证明了 ENDWEAVER 的有效性和稳健性，但仍存在若干局限性，有必要对此进行讨论，并为未来的改进提供方向。

首先，ENDWEAVER 中的多层级规划高度依赖于任务描述的可用性。如果任务描述含糊不清、不完整，或未能明确体现存在多个结局，系统探索多样化路径的能力就会受到显著限制。因此，提供高质量且结构良好的任务描述，是释放 ENDWEAVER 全部潜力的基本前提。

其次，本研究所采用的任务相对普通，主要依赖基础逻辑和常识知识——而这些内容在通用大语言模型的预训练过程中都得到了充分覆盖。这有助于系统在此类设置下保持稳定性能。然而，当应用于更复杂或更具领域特定性的场景时，底层大语言模型的推理能力可能成为瓶颈，从而对整体性能产生负面影响。除能力限制之外，成本也是另一项重要考量。尽管我们的实验主要报告了环境交互成本，但 ENDWEAVER 的实际运行时成本还包括大语言模型调用所带来的计算开销。换言之，交互效率并不能完全转化为总体成本效率。在当前评估中，每个任务的平均输入词元用量在 Overcooked Plus 中约为 42k ± 9k，在 Minecraft 中约为 44k ± 13k，在 PySC2 中约为 58.5k ± 20.8k；相应的平均输出词元用量分别约为 7k ± 1k、8.5k ± 1.8k 和 6.5k ± 1.8k。这一推理成本很可能会随着游戏任务时长增加、机制更加丰富以及分支结构更加复杂而进一步上升。此外，我们的初步实验表明，在 PySC2 中以在线生成替代离线原始任务生成，可能会使每个任务的输入词元消耗增加至约 461.2k ± 320.8k，从而形成明显的性能—成本权衡。由于性能和成本都与基础模型的质量及其与任务的匹配程度密切相关，未来的部署必须在二者之间取得

平衡。例如，可以使用经过领域特定数据微调、在本地部署的较小模型，以降低推理成本，同时保持足够的规划能力。

第三，随着游戏和任务复杂度的增加，必须对状态—动作转移图进行抽象，以避免处理冗余或过于细粒度的状态细节，同时保留关键节点及其连通性。然而，当交互对象和任务状态的数量急剧增长时，如果未采用有效的状态等价抽象机制，图结构的规模可能迅速扩大。这可能导致存储、检索和采样效率显著下降，给系统带来更大的可扩展性挑战。

F. 有效性威胁

对于内部有效性，首先值得关注的是大语言模型输出固有的随机性，这可能影响结果的可复现性和性能稳定性。为此，我们在相同设置下多次进行每项测试，并报告均值及95%置信区间。其次，大语言模型的预训练数据并不透明，且可能包含与某些任务或环境重叠的知识，从而引入潜在的评估偏差。为降低这种影响，我们使用来自不同厂商的多个大语言模型（例如 GPT-4o、通义千问和 DeepSeek），以减少对任何单一先验知识来源的依赖。第三，基线方法的性能可能受到奖励函数设计和超参数的影响，而欠佳的配置可能导致不公平的比较。为尽量降低这一风险，我们按照最佳实践设计了具有竞争力的奖励函数和参数设置。

关于外部效度，一个关键问题是评估环境范围受限。我们在四款游戏——Mini RPG、Overcooked Plus、Minecraft 和基于 PySC2 的即时战略场景——中进行测试，这些游戏旨在代表多样化的游戏类型和任务结构。然而，它们仍无法完全反映现实世界游戏的广泛多样性，例如第一人称射击游戏、益智游戏或叙事性较强的冒险游戏。未来工作需要开展更多评估，以验证方法的泛化能力。另一个问题涉及任务描述的真实性。尽管我们参考实际商业游戏进行任务设计，并融入了多结局机制，但这些描述本身仍是人工构建的。现实世界中的游戏任务——尤其是在工业场景中——可能包含更多噪声、结构性更弱或经过有意混淆的信息。

VI. 结论与未来工作

本文介绍了一种基于大语言模型的新型框架 ENDWEAVER，用于对多结局游戏任务进行自动化冒烟测试。该框架将多层级规划与动态状态-动作转移图相结合，有效应对了多结局游戏冒烟测试中的路径探索与效率挑战。在多层级规划中，高层方法确保对任务路径进行广泛探索，而低层原始任务则提供更细粒度的规划单元，从而实现更精准的任务执行和更全面的多样化游戏玩法场景覆盖。动态状态-动作转移图记录了任务执行期间状态与动作之间的关系，在消除路径冗余的同时支持全面的路径覆盖，并为后续测试奠定基础。

在实验评估中，我们在四种不同的游戏环境中评估了 ENDWEAVER 框架的有效性：Mini RPG 、Overcooked Plus、Minecraft 和 PySC2。我们将 ENDWEAVER 与四种基线方法进行了比较——随机方法、遗传算法、近端策略优化和 ELLM-PPO。结果表明，ENDWEAVER 在任务完成率、结局多样性和游戏交互效率方面均优于这些方法。此外，我们还开展了消融研究，以评估多层级规划对全面探索的贡献，以及图结构在路径生成中的作用。进一步地，通过比较不同的大语言模型，我们确认了框架性能与所使用大语言模型能力之间的正相关关系。

尽管该框架已展现出良好的效果，但仍需在以下方向上进一步探索和改进。首先，未来工作的重点将是与行业合作伙伴开展密切合作，在实际工业场景中评估该框架的有效性。目前实验所使用的环境仍相对简单，在真实工业场景中评估该框架将有助于检验其实际适用性，并确保其能够适用于更复杂的现实游戏开发环境。其次，对动态环境的支持也是未来发展的一个方向。当前的 ENDWEAVER框架主要面向相对静态的环境而设计。然而，在更复杂的工业游戏，尤其是在线游戏中，游戏对象和状态可能会发生不可预测的变化。因此，未来的工作将重点整合实时监控和自适应规划策略。这将确保该框架即使在高度动态且不可预测的环境中，也能维持稳定而有效的测试。最后，我们计划将 ENDWEAVER扩展为面向工程师的人在回路测试助手。由于该框架已经能够生成结构化中间输出以及可解释的状态—动作转移图，这些结果可以导出为 XML 等结构化格式，并与面向用户的工具集成。一种可能的界面可以提供基于图的已探索状态和结局可视化、可编辑任务计划，以及将已验证轨迹一键导出为工程产物的功能。该方向有望通过帮助测试工程师更有效地指导、检查和复用自动化测试结果，进一步提升该框架的实际可用性。

致谢

本研究得到了 JST SPRING（项目编号：JPMJSP2128）和 JSPS KAKENHI（项目编号：23K28064、25K15290）的资助。

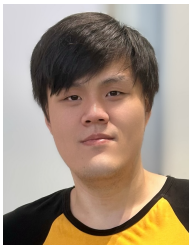
参考文献

[1] S. V¨artinen, P. H¨am¨al¨ainen 和 C. Guckelsberger，“使用 GPT 语言模型生成角色扮演游戏任务”，*IEEE Transactions on Games*，第 16 卷，第 1 期，第 127–139 页，2022年。

[2] G. Smith、R. Anderson、B. Kopleck、Z. Lindblad、L. Scott、A. Wardell、J. Whitehead 和 M. Mateas，“任务定位：角色扮演游戏中任务与关卡设计的设计模式”，载于《交互式数字叙事国际会议》。施普林格，2011年，第 326–329 页。[3] Y. Wu、Y. Chen、X. Xie、B. Yu、C. Fan 和 L. Ma，“大型多人在线角色扮演游戏的回归测试”，载于国际软件维护与演化会议。IEEE，2020年，第 692–696 页。[4] S. McConnell，“每日构建与冒烟测试”，IEEE Software，第 13 卷，第 4 期，第 144 页，1996年。[5] C. Cannavacciuolo 和 L. Mariani，“云系统的冒烟测试”，载于 2022 IEEE软件测试、验证与确认会议（ICST），2022年，第 47–57 页。[6] J. Hu、M. Zhang、B. Liu、Y. Wu 和 Y. Chen，“一种用于游戏任务冒烟测试的语言引导加速方法”，载于第 35 届 IEEE软件可靠性工程国际研讨会 ISSRE 2024——研讨会，筑波，日本，2024年10月28日至31日。IEEE，2024年，第 15–20 页。[7] S. Herbold 和 T. Haar，“机器学习冒烟测试：发现严重缺陷的简单测试”，《实证软件工程》，第 27 卷，第 2 期，第 45 页，2022年1月。[8] C. Politowski、F. Petrillo 和 Y.-G. Gu´eh´eneuc，“视频游戏测试综述”，载于 2021 IEEE/ACM 软件测试自动化国际会议（AST）。IEEE，2021年，第 90–99 页。[9] A. Memon、I. Banerjee、N. Hashmi 和 A. Nagarajan，“Dart：用于图形用户界面应用程序‘夜间/每日构建’回归测试的框架”，载于国际软件维护会议，2003年。ICSM 2003 会议录。IEEE，2003年，第 410–419 页。[10] C.-S. Cho、D.-C. Lee、K.-M. Sohn、C.-J. Park 和 J.-H. Kang，“在线游戏服务器黑盒负载测试的基于场景的方法”，载于 2010年网络赋能分布式计算与知识发现国际会议，2010年，第 259–265 页。[11] M. Ostrowski 和 S. Aroudj，“电子游戏开发中的自动化回归测试”，GSTF 计算期刊（JoC），第 3 卷，第 2 期，第 10 页，2013年。[12] S. Iftikhar、M. Z. Iqbal、M. U. Khan 和 W. Mahmood，“一种用于平台游戏的基于模型的自动化测试方法”，载于 2015 ACM/IEEE 第 18 届模型驱动工程语言与系统国际会议（MODELS），2015年，第 426–435 页。[13] J. Hern´andez B´ecares、L. Costero Valero 和 P. P. G´omez Mart´ın，“一种自动化电子游戏测试方法”，《娱乐计算》，第 18 卷，第 79–92 页，2017年。 [在线]。可访问：https: //www.sciencedirect.com/science/article/pii/S1875952116300254 [14] J. Bergdahl、C. Gordillo、K. Tollmar 和 L. Gisslen，“使用深度强化学习增强自动化游戏测试”，载于 2020 IEEE游戏会议（CoG），2020年，第 600–603 页。[15] A. Sestini、L. Gissl´en、J. Bergdahl、K. Tollmar 和 A. D. Bagdanov，“利用好奇心条件近端轨迹进行自动化游戏玩法测试与验证”，IEEE Transactions on Games，第 16 卷，第 1 期，第 113–126 页，2024年。[16] C. Gordillo、J. Bergdahl、K. Tollmar 和 L. Gissl´en，“通过好奇心驱动强化学习智能体提升游戏测试覆盖率”，载于 2021 IEEE游戏会议（CoG），2021年，第 1–8 页。[17] Y. Zheng、X. Xie、T. Su、L. Ma、J. Hao、Z. Meng、Y. Liu、R. Shen、Y. Chen 和 C. Fan，“Wuji：使用进化深度强化学习进行自动化在线战斗游戏测试”，载于 2019年 IEEE/ACM 第 34 届自动化软件工程国际会议（ASE）。IEEE，2019年，第 772–784 页。[18] C. Holmg˚ard、M. C. Green、A. Liapis 和 J. Togelius，“通过基于演化启发式的蒙特卡洛树搜索，利用程序化角色进行自动化游戏测试”，IEEE Transactions on Games，第 11 卷，第 4 期，第 352–362 页，2019年。[19] W. Ahlberg、A. Sestini、K. Tollmar 和 L. Gissl´en，“使用多模态对抗模仿学习生成游戏角色”，载于 2023 IEEE游戏会议（CoG），2023年，第 1–8 页。[20] S. Ariyurek、A. Betin-Can 和 E. Surer，“使用合成类人智能体进行自动化电子游戏测试”，IEEE Transactions on Games，第 13 卷，第 1 期，第 50–67 页，2021年。[21] Y. Duarte、H. C. Mandelli、V. H. S. Durelli、P. A. Nardi 和 A. T. Endo，“平台电子游戏的探索性测试：策略与经验教训”，J. Interact. Syst.，第 15 卷，第 1 期，第 657–669 页，2024年。[22] M. Zhang、X. Zhang、P. Arcaini 和 F. Ishikawa，“对围棋中机器学习智能体行为的研究”，载于 DSA。IEEE，2023年，第 734–742 页。[23] P. Spronck、M. Ponsen、I. Sprinkhuizen-Kuyper 和 E. Postma，“通过动态脚本实现自适应游戏人工智能”，《机器学习》，第 63 卷，第 3 期，第 217–248 页，2006年。

[24] I. Zarembo，《人工智能在电子游戏自动化测试中的应用分析》，载于《第12届国际科学与实践会议论文集》第II卷，第170卷，2019年，第174页。[25] A. Zook、E. Fruchter 和 M. O. Riedl，《通过主动学习自动进行游戏测试以调优游戏参数》，arXiv 预印本 arXiv:1908 01417，2019年。[26]S. Ariyurek、A. Betin-Can 和 E. Surer，《改进用于电子游戏测试的蒙特卡洛树搜索算法》，arXiv 预印本 arXiv:2003.07813，2020年。[27] D. Loubos，《虚拟世界中的自动化测试》，硕士论文，2018年。[28] J. Pfau、J. D. Smeddinck 和 R. Malaka，《借助 Icarus 自动进行游戏测试：通过无监督求解智能完成冒险谜题》，载于《年度计算机与人类互动游戏研讨会扩展摘要集》，2017年，第153–164页。[29] S. Roohi、J. Takatalo、C. Guckelsberger 和 P. H¨am¨al¨ainen，《基于模拟的游戏测试中的内在动机综述》，载于《2018年CHI计算系统中的人为因素会议论文集》，2018年，第1–13页。[30] C. Guerrero-Romero、S. M. Lucas 和 D. Perez-Liebana，《使用一组通用人工智能算法辅助游戏设计与测试》，载于《2018年IEEE计算智能与游戏会议（CIG）》。IEEE，2018年，第1–8页。[31] P. Ammanabrolu 和 M. O. Riedl，《使用基于图的深度强化学习玩文字冒险游戏》，arXiv 预印本 arXiv:1812.01628，2018年。[32] K. Debnath、R. Omar、N. Abdul Latip、S. Shelyna 等，《面向无人飞行器最优节能路径规划的图搜索算法综述》，《印度尼西亚电气工程与计算机科学期刊》，第15卷，第2期，第743–749页，2019年。[33] P. Auer、N. Cesa-Bianchi 和 P. Fischer，《多臂老虎机问题的有限时间分析》，《机器学习》，第47卷，第2–3期，第235–256页，2002年。[34] D. Silver、J. Schrittwieser、K. Simonyan、I. Antonoglou、A. Huang、A. Guez、T. Hubert、L. Baker、M. Lai、A. Bolton 等，《在没有人类知识的情况下精通围棋》，《自然》，第550卷，第7676期，第354页，2017年。[35] A. Joulin、E. Grave、P. Bojanowski 和 T. Mikolov，《高效文本分类的技巧集》，载于《第15届计算语言学协会欧洲分会会议论文集：第2卷，短篇论文》，计算语言学协会，2017年4月，第427–431页。[36] C. C. Aggarwal 和 C. Zhai，《文本分类算法综述》，载于《文本数据挖掘》，施普林格，2012年，第163–222页。[37] J. He、J. Chen、X. He、J. Gao、L. Li、L. Deng 和 M. Ostendorf，《在自然语言动作空间中进行深度强化学习》，arXiv 预印本 arXiv:1511.04636，2015年。[38] D. L. Chen 和 R. J. Mooney，《从观测中学习理解自然语言导航指令》，载于第二十五届 AAAI 人工智能会议，2011年。[39] J. Hu、M. Zhang、B. Liu、Y. Wu 和 Y. Chen，《一种用于游戏任务冒烟测试的语言引导加速方法》，载于《2024年IEEE第35届软件可靠性工程国际研讨会专题研讨会（ISSREW）》。IEEE，2024年，第7–12页。[40] G. Wang、Y. Xie、Y. Jiang、A. Mandlekar、C. Xiao、Y. Zhu、L. Fan 和 A. Anandkumar，《Voyager：一种基于大语言模型的开放式具身智能体》，2023年。 [在线]。可访问：https://arxiv.org/abs/2305 16291 。H. Warpefelt 和 H. Verhagen[41]，《非玩家角色可信度模型》，《游戏与虚拟世界期刊》，第9卷，第1期，第39–53页，2017年。[42] T. Rist，《使用大语言模型将虚拟3D世界的探索转化为互动叙事体验》，载于《2024年IEEE游戏会议（CoG）》。IEEE，2024年，第1–8页。[43] N. Ranella 和 M. Eger，《利用生成式人工智能自动生成电子游戏解说》，载于 EXAG@AIIDE，2023年。[44] X. You、P. Taveekitworachai、S. Chen、M. Can Gursesli、X. Li、Y. Xia 和 R. Thawonmas，《地牢、龙与情感：对大语言模型驱动的桌面角色扮演游戏中玩家情感的初步研究》，载于《第19届数字游戏基础国际会议论文集》，2024年，第1–4页。[45] T. Wang、M. Honari-Jahromi、S. Katsarou、O. Mikheeva、T. Pana-giotakopoulos、S. Asadi 和 O. Smirnov，《player2vec：一种理解游戏中玩家行为的语言建模方法》，arXiv 预印本 arXiv:2404.04234，2024年。[46] G. Todd、S. Earle、M. U. Nasir、M. C. Green 和 J. Togelius，《通过大语言模型生成关卡》，载于第18届

数字游戏基础国际会议，2023年，第1–8页。[47] E. Decart、Q. McIntyre、S. Campbell、X. Chen 和 R. Wachen，《Oasis：变换器中的宇宙》，网址：https://oasis-model. github. io，2024年。[48] S. Reed、K. Zolna、E. Parisotto、S. G. Colmenarejo、A. Novikov、G. Barth-Maron、M. Gimenez、Y. Sulsky、J. Kay、J. T. Springenberg 等，《通用智能体》，arXiv 预印本 arXiv:2205.06175，2022年。[49] Z. Wang、S. Cai、A. Liu、Y. Jin、J. Hou、B. Zhang、H. Lin、Z. He、Z. Zheng、Y. Yang 等，《JARVIS-1：基于记忆增强多模态语言模型的开放世界多任务智能体》，arXiv 预印本 arXiv:2311.05997，2023年。[50] H. Yuan、C. Zhang、H. Wang、F. Xie、P. Cai、H. Dong 和 Z. Lu，《面向开放世界长时程任务的技能强化学习与规划》，arXiv 预印本 arXiv:2303.16563，2023年。[51] K. Nottingham、P. Ammanabrolu、A. Suhr、Y. Choi、H. Hajishirzi、S. Singh 和 R. Fox，《具身智能体会梦见像素化绵羊吗？使用语言引导的世界建模进行具身决策》，载于《国际机器学习会议》。PMLR，2023年，第26 311–26 325页。[52] Y. Du、O. Watkins、Z. Wang、C. Colas、T. Darrell、P. Abbeel、A. Gupta 和 J. Andreas，《利用大语言模型引导强化学习预训练》，载于《国际机器学习会议》。PMLR，2023年，第8657–8677页。[53] S. Yao、J. Zhao、D. Yu、N. Du、I. Shafran、K. Narasimhan 和 Y. Cao，《ReAct：在语言模型中协同推理与行动》，载于《国际学习表征会议（ICLR）论文集》。ICLR，2023年，第7566–7580页。[54] S. Yao、D. Yu、J. Zhao、I. Shafran、T. L. Griffiths、Y. Cao 和 K. Narasimhan，《思维树：使用大语言模型进行审慎的问题求解》，载于《神经信息处理系统进展（NeurIPS）论文集》。NeurIPS，2023年，第55 550–55 564页。[55] X. Wang、Y. Chen、L. Yuan、Y. Zhang、Y. Li、H. Peng 和 H. Ji，《可执行代码操作可激发更优秀的大语言模型智能体》，载于《国际机器学习会议（ICML）论文集》。ICML，2024年。[56] J. Wei、X. Wang、D. Schuurmans、M. Bosma、b. ichter、F. Xia、E. Chi、Q. V. Le 和 D. Zhou，《思维链提示可激发大语言模型的推理能力》，载于《神经信息处理系统进展》，S. Koyejo、S. Mohamed、A. Agarwal、D. Belgrave、K. Cho 和 A. Oh 编，第35卷，2022年，第24 824–24 837页。[57] N. Shinn、F. Cassano、A. Gopinath、K. Narasimhan 和 S. Yao，《Reflexion：采用语言强化学习的语言智能体》，载于《神经信息处理系统进展》，第36卷。Curran Associates, Inc.，2023年，第8634–8652页。[在线]。可访问：https://proceedings.neurips.cc/paper files/paper/2023/file/1b44b878bb782e6954cd888628510e90-Paper-Conference.pdf [58] Z. Li、Y. Wu、L. Ma、X. Xie、Y. Chen 和 C. Fan，《GBGallery：游戏测试的基准测试与框架》，《实证软件工程》，第27卷，第6期，第140页，2022年。[59] J. H. Holland，《遗传算法》，《科学美国人》，1992年7月。[60] J. Schulman、F. Wolski、P. Dhariwal、A. Radford 和 O. Klimov，–《近端策略优化算法》，2017年。 [在线]。可访问：https://arxiv.org/abs/1707 06347 。[61] Z. Li、Y. Ni、R. Qi、L. Jiang、C. Lu、X. Xu、X. Liu、P. Li、Y. Guo、Z. Ma、H. Li、H. Wu、X. Guo、K. Huang 和 X. Zhang，《LLM-PySC2：面向大语言模型的星际争霸 II 学习环境》，2025年。[ ] 2411 05348 在线。可访问：https://arxiv.org/abs/。[62] J. Cai、J. Li、N. Li、M. Zhang、R. Yang 和 K. Tei，《Overcooked Plus：用于提升自主规划算法评估水平的综合烹饪场景测试平台》，载于《第5届IEEE自主计算与自组织系统国际会议（ACSOS）论文集》，2024年9月。[63] Ghost Town Games Ltd.，《胡闹厨房！》Team17，2024年，访问日期：2024-08-26。 [在线]。可访问：https://www.team17.com/games/overcooked/ [64] Mojang Studios，《Minecraft》，https://www.minecraft.net，2009年，由Mojang Studios开发的沙盒电子游戏。[65] O. Vinyals、T. Ewalds、S. Bartunov、P. Georgiev、A. S. Vezhnevets、M. Yeo、A. Makhzani、H. K¨uttler、J. Agapiou、J. Schrittwieser、J. Quan、S. Gaffney、S. Petersen、K. Simonyan、T. Schaul、H. van Hasselt、D. Silver、T. Lillicrap、K. Calderone、P. Keet、A. Brunasso、D. Lawrence、A. Ekermo、J. Repp 和 R. Tsing，《星际争霸 II：强化学习的新挑战》，2017年。 [在线]。可访问：https://arxiv.org/abs/1708.04782



Jinyu Cai 目前正在日本东京的早稻田大学攻读计算机科学博士学位，并获得 JST SPRING 资助。他于2023年获得早稻田大学计算机科学工学硕士学位。他的研究兴趣包括自主智能体和大型语言模型。



Mingyue Zhang 是中国重庆西南大学的副教授。他于2017年获得中国大连大连理工大学的计算机科学工学学士学位，并于2023年获得中国北京北京大学的计算机科学博士学位。他的研究兴趣包括自适应系统、多智能体系统和强化学习。



Nianyu Li 是一名专门研究人类参与型自适应系统的研究人员。她于2021年在 Zhi Jin 教授和 Wenpin Jiao 教授的指导下，获得北京大学计算机软件与理论博士学位。她还曾在国立信息学研究所（NII）跟随 Zhenjiang Hu 教授，以及在卡内基梅隆大学（CMU）跟随 David Garlan 教授开展研究实习。她的研究兴趣包括软件建模与分析、需求工程、规约与验证、系统安全，信息安全和信息物理系统。



Jinglue Xu 是 Sakana AI 的研究员。他分别于2022年 和 2025年 获得东京大学计算机科学理学硕士和博士学位。他的研究兴趣包括大语言模型、自主智能体、演化计算和自动化机器学习，尤其专注于开发自动化、协作性强且易于使用的人工智能系统。



Tianshui Li 目前正在日本东京的东京大学信息科学与技术研究生院信息与通信工程系攻读博士学位。他于2019年在日本会津若松的会津大学获得计算机科学与工程理学学士学位。他的研究兴趣包括进化计算、群体智能、大语言模型和数值优化。



Jialong Li 是日本东京早稻田大学的助理教授，同时也是东京科学大学的研究员。他于2019年、2021年和2025年分别获得早稻田大学计算机科学工学学士、工学硕士和博士学位。他的研究兴趣包括自适应系统和需求工程。他是 IEEE、ACM、IPSJ 和 IEICE 的成员。



Hitoshi Iba 是日本东京大学信息科学与技术研究生院信息与通信工程系教授。他于1990年获得东京大学信息工程博士学位。同年，他加入电气技术实验室，并于1996年至1997年在美国加利福尼亚州斯坦福的斯坦福大学担任访问研究员。

他于1998年加入东京大学信息与通信工程系，担任副教授，并于2004年晋升为教授。他的研究兴趣包括进化系统、人工生命和人工智能基础。



Kenji Tei 是日本东京科学大学副教授。他还在早稻田大学和日本东京的国立信息学研究所担任特别任命的副教授。他于2008年获得早稻田大学工程学博士学位。他的研究兴趣包括自适应系统、软件架构和模型驱动工程。他是IEEE计算机学会、ACM、IPSJ和IEICE的成员，并担任《ACM自主与

自适应系统。