

Agent Planning & Task Decomposition

智能体规划与任务分解

2026年5月28日 | AI Agent 知识分享 | 第8期

一、一句话概括

智能体规划与任务分解 (Planning & Task Decomposition) 是Agent将复杂目标自动拆解为可执行的子任务序列的能力——从「我要做什么」到「我先做什么、再做什么」的系统化思考过程，是Agent从简单问答进化为自主完成多步骤复杂任务的核心能力，也是区分「工具调用」和「自主Agent」的关键分水岭。

二、核心原理

>> 自上而下规划 (Top-Down Planning)

Agent在行动前对整个任务进行完整分解，生成树状子任务结构 (Task Tree)，然后按顺序或依赖关系依次执行。典型代表为 Hierarchical Task Network (HTN) 和 Plan-and-Solve (PS) 方法。优点是计划全局可控，缺点是难以应对执行过程中的意外变化。适合结构化程度高的任务（如软件部署流程、数据处理流水线）。

>> 自下而上规划 (Bottom-Up Planning)

Agent不预先做完整的全局规划，而是在执行过程中动态决策下一步做什么。每一步基于当前观察到的结果「即兴发挥」。典型代表为 ReAct (思考-行动-观察循环) 和 Tree-of-Thoughts (多路径探索)。适合开放性强、不可预测的任务（如网页导航、客服对话）。缺点是可能陷入局部最优或遗漏关键步骤。

>> 混合规划 (Hybrid Planning)

先做高层次的粗粒度规划 (High-Level Plan)，然后对每个子任务在执行时再做细粒度的动态规划。结合了自上而下的全局视角和自下而上的灵活性。典型代表为 LLM+P (用PDDL形式化语言做高层规划)、SayCan (机器人领域的可行方案筛选)。这是当前最先进的Agent规划方案，兼顾效率与灵活性。

核心要点：规划的粒度 (Granularity) 是关键——太粗导致步骤模糊无法执行，太细导致Token浪费和组合爆炸
规划与执行之间需要有反馈循环：执行结果影响下一步规划，形成「计划→执行→反馈→调整→再计划」的闭环
好的规划系统应该能在过规划 (Over-planning) 和欠规划 (Under-planning) 之间找到平衡点

三、测试实战

测试一：规划覆盖率测试 (Coverage Rate Test)

操作：设计一个需要5-8步的复合任务（如「从网页抓取数据→清洗→分析→生成报告→发送邮件」），给Agent但不说具体步骤，看Agent能否分解出所有必要子任务，是否遗漏关键环节

工具：LangSmith Hub (<https://smith.langchain.com/hub>) 用于追踪规划轨迹，或使用 Open AI Playground (<https://platform.openai.com/playground>) 直接测试 GPT-4o 的规划能力

指标：规划覆盖率 (Coverage Rate) = 分解出的子任务数 / 实际需要的子任务数；遗漏率 (Missing Rate) = 遗漏的关键步骤数

测试二：规划质量与效率评估 (Quality vs Efficiency Test)

操作：对同一个任务（如「写一篇行业分析报告」），分别用 Top-Down、Bottom-Up、Hybrid 三种规划方式让Agent执行，对比完成率、执行步数、Token总消耗

工具：使用 LangChain (<https://www.langchain.com>) 的 AgentExecutor 分别配置不同 Planner；或用 AutoGPT (<https://github.com/Significant-Gravitas/AutoGPT>) 的 planner 模块

指标：任务完成率 (Completion Rate)、每步平均耗时 (Avg Latency/Step)、Token效率 (Tokens per Completed Task)、规划体积 (Plan Tokens / Total Tokens 的比例应低于30%)

测试三：动态重规划能力测试 (Replanning Robustness Test)

操作：Agent正在按计划执行时，中途改变外部条件（如「已经抓取的数据库突然不可访问」「审批预算从10万改为5万」），观察Agent能否自动调整原计划而不是卡死或从头开始

工具：自定义Mock Server模拟外部环境变化；或使用 CrewAI (<https://www.crewai.com>) 的 Process 模块做规划切换实验

指标：重规划成功率 (Replanning Success Rate)、从干扰到恢复的时间 (Recovery Latency)、原计划修改比例 (Plan Modification Ratio, 越高说明灵活性越好)

测试四：规划幻觉与可行性测试 (Planning Hallucination Test)

操作：观察Agent是否规划了「依赖不存在/无权访问的工具」或「逻辑上不可能并行执行的任务」，例如分解出一个「调用内部HR系统的API」但Agent根本没有这个工具

工具：自定义Agent日志分析脚本，提取所有规划中的工具调用声明，与实际工具注册表做交叉验证

指标：规划幻觉率 (Planning Hallucination Rate) = 不可行子任务数 / 总子任务数；工具可用性匹配率 (Tool Availability Match Rate)

四、常见坑点

坑点1：过规划（Over-planning）—— 用大量Token做完美计划，结果根本执行不了

弱模型或过度优化的规划器会生成极其详细的子任务树（10+步骤），但大部分步骤从未被执行——Agent花了80%的Token做计划，只花了20%执行。典型的「计划得天花乱坠，执行时寸步难行」。解决方案：设置规划深度限制（max_plan_depth=3~5），规划Token占比监控预警（超过总Token30%触发自动简化），以及规划即执行（执行完当前步骤才规划下一步，而非一步到位）。

坑点2：规划与执行脱节——计划做了一整套，Agent实际干的是另一套

这是最常见的系统性失败模式：Agent生成了完美的计划，但在执行第一步后就偏离了原计划走向，计划变成了「文档装饰品」。根因是规划器（Planner）和执行器（Executor）之间的反馈回路断裂——Executor没有把执行遇到的困难反馈给Planner让后者调整计划。解决方案：引入Plan-Tracking机制，每一步执行后比较「当前结果 vs 计划预期」，发现偏离超过阈值则触发Replan。

坑点3：任务粒度失控——分解太粗糙或太细致都导致失败

任务分解的粒度是规划系统最棘手的超参数。太粗（如「写一份报告」作为一步）让Executor无从下手；太细（如「移动鼠标到坐标(100,200)→点击→等待200ms→移动鼠标到坐标(300,400)」）导致组合爆炸。好的做法是动态粒度：高层计划用粗粒度，执行到具体子任务时再自动细粒度分解。实践中可以设置「子任务不可再分解」的条件（如「子任务可以独立在5步内执行完」）。

五、延伸阅读

- Plan-and-Solve 论文 (Wang et al. 2023) : <https://arxiv.org/abs/2305.04091>
- Tree-of-Thoughts (Yao et al. 2023) : <https://arxiv.org/abs/2305.10601>
- LLM+P (Liu et al. 2023) : <https://arxiv.org/abs/2304.11477>
- Anthropic 关于 Building Effective Agents: <https://docs.anthropic.com/en/docs/build-with-claude/agent>
- SayCan (Google Robotics, 2022) : <https://saycan.github.io/>
- LangGraph 规划模块文档: <https://langchain-ai.github.io/langgraph/tutorials/plan-and-execute/>