*github：*https://github.com/keta1930/mcp-agent-graph

*sdk docs：*https://keta1930.github.io/mcp-agent-graph/#

MIT License

发布时间：2025-04-24

前言：mcp-agent-graph 是一款高效、轻量、易上手的 Agent 开发框架。

作为 Agent 开发工作者，您可能会使用类似Dify，扣子等可视化工作流编排；您可能更习惯代码开发，那么您可能接触的是langgraph、crewai、Google ADK等等。

mcp-agent-graph 和上述框架有何不同？有什么值得推荐的理由？

本文将介绍项目的设计理念，功能特点，未来发展规划。用最短的时间让您了解这个项目，或许您会发现这个项目正符合您的需求。

1 agent graph

项目基于“图”来实现 agent 构建。图由 node （节点）构成。以下图为例：图1 为循环图 ；图2 为有向无环图。

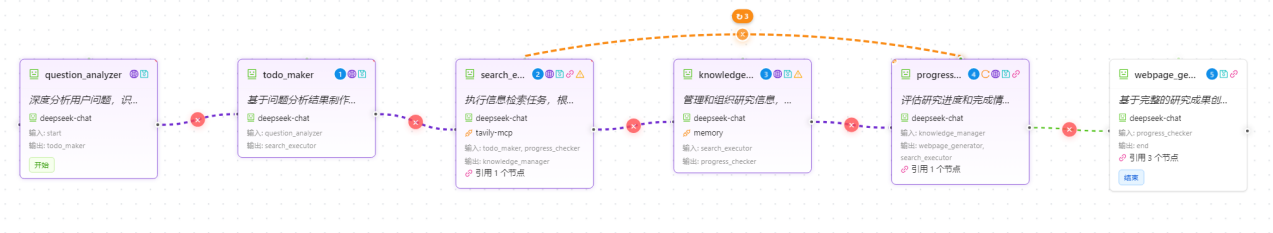

图 1 循环图 - 项目示例

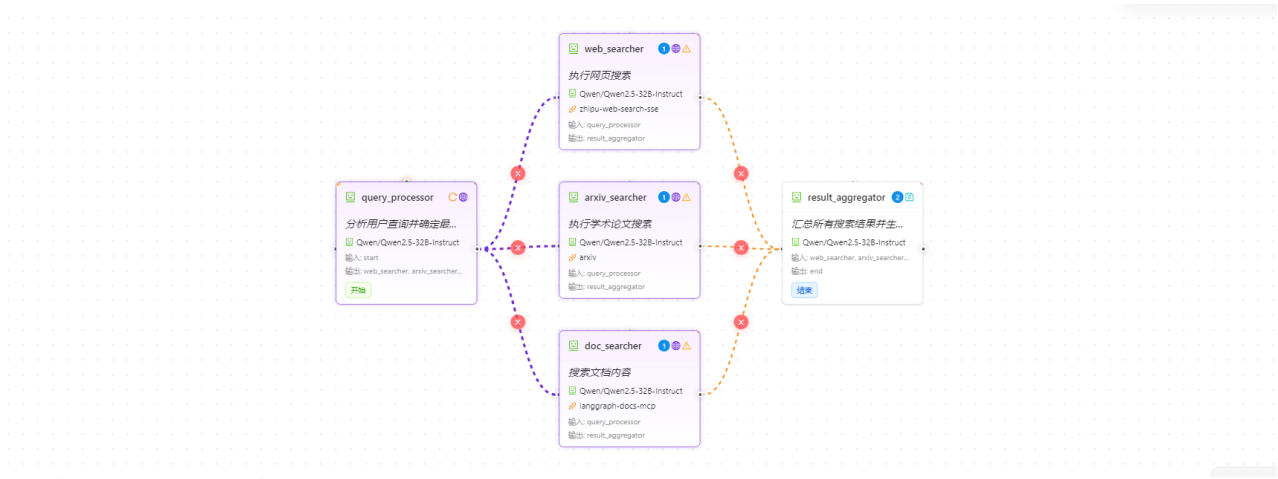

图 2 有向无环图 -项目示例

Q：图 如何构建?

本项目有**三种方式**快速构建一张图：

（1）AI 生成

AI 生成是最快的一种方式，项目提供了提示词，帮助 LLM 理解如何作图。您只需要提供需求，剩下都由 AI 完成。

图 3 AI 生成图 - 步骤1

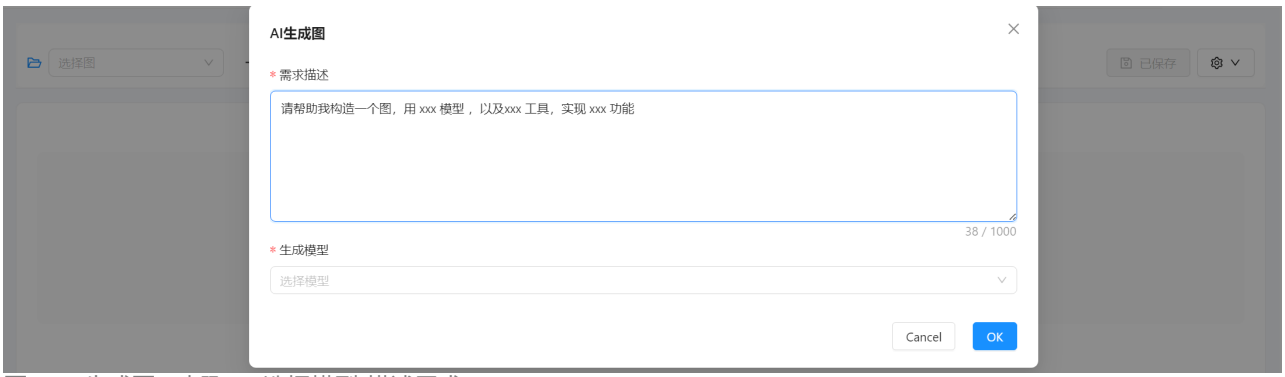

图 4 AI 生成图 - 步骤2：选择模型-描述需求

您也可以**复制提示词，到其他 AI 平台生成图的配置（.json）**，然后一键导入这张图。

图 5 提示词模版 - 里面包含了当前系统中注册的模型和 MCP 服务；以及详细的作图指南

图 6 导入JSON 图

通过 **AI 生成**的方式，您就快速得到了一张符合您需求的图。

（2）前端作图

您也可以直接在前端新建图，增加节点，配置 工具、提示词、循环...并编排流程。这和Dify相似。

（3）合作生图

假如您是团队中的一员，您只需要完成大图当中的一环，那么您可以在制作完成图以后，**将图打包。**

图 7 导出图为压缩包

压缩包内，会包含**图的配置文件（.json）/readme.md**（项目将自动生成readme文件，无需您手动编写）/**提示词文件**（本项目支持在作图时，提示词引用本地文档： 如：user_prompt: "严格遵循：{xxx.md}"，文档在生成图的时候会被保存一份副本到 图的文件夹当中，并在打包的时候一并打包。）/**模型配置**（将图使用到的模型提取并打包）/**MCP配置**（将图使用到的MCP服务提取并打包）

其他项目成员可以将**压缩包导入**，直接使用您的成果。打通运行流程。

TODO：每一个节点将被设计为可复制的。实现从一个 图 当中复制 （某一个/一批）节点 到另一个 图 当中。

2 node

节点 是图的核心。

图 由节点构成，一个 图 可以是**单节点** ， 也可以是**多节点。**

节点 可以是一个**模型+mcp** ， 也可以是**一张图 （子图概念）**

Q：**节点** 包含哪些功能?

节点有以下参数可配置：

（1）节点名称

（2）节点描述

（3）节点类型

可以是**llm / graph**。如果选择graph，那么可以选择已有的**图 作为节点**

（4）模型

如果节点类型为llm ， 则可以选择系统当中已注册的模型

（5）mcp 服务 （可用工具/资源）

（6）输入节点

（7）输出节点

（8）启用输出/禁止输出

当模型调用mcp 工具时，有两类情况：

LLM -> mcp -> Tool result -> LLM -> answer -> output node （第一种情况对应于启用输出）

LLM -> mcp -> Tool result -> output node（第二种情况对应于禁用输出）

（9）提示词

可以使用{node_name}的形式引用 上游节点的信息。

可以使用{xxx.md/doc}的形式引用 本地文件当中的提示词。并会将提示词文件 copy 副本到图文件夹当中

例如以下示例：

```
message:
[
系统提示词： 严格遵循以下要求：{E:\可视化.txt}
用户提示词： 将 {search node} 获取得到的内容按照要求进行可视化。
]
```

（10）全局输出

在正常情况下：

一个节点的提示词只能包含 其**输入节点**的内容

例如：

start -> node1 - > node2 -> node3 -> node4 -> node2/end

node3 可以引用 node2 的输出结果，但是没法引用node1 或者是node4 的输出结果。

开启全局输出，那么节点的输出将实现全局可引用、

并且在**循环的图**当中，全局输出的节点 **每一轮执行的结果都会被保留下来。**其余节点可以选择性调用 该节点 的 **全部轮数的输出/最新一轮的输出/最新X轮的输出**

（11）handoffs

handoffs 借鉴了openai agent sdk 的handoffs功能

该参数的设计理念是：

将 **输出节点** 包装成工具，选择 哪一个工具，就会**把任务交接到那个节点。**

handoffs 参数使得项目**能够制作循环的图**

为了确保handoffs顺利，在节点描述/提示词当中可进行增强。

例如可以在提示词当中说明：

```
你需要调用工具完成任务，如果xxx发生，你需要选择x继续...
否则你需要选择y进行...
```

（12）保存格式

如果您希望将节点输出的结果保存为某一个文件，可以**选择扩展名**：如 md html doc txt py...

在运行过程中，节点输出以后会把**输出附件**保存到**结果文件夹**当中

（13）引用节点

配合全局输出功能，实现对全局输出节点的引用。

图 8 节点参数配置 - 基础信息

3 mcp

mcp 是节点的核心

开发者认为：

越来越多的API 将会转为 mcp 服务

MCP正在成为**AI工具集成的行业标准**，越来越多的工具提供商开始提供MCP版本。这意味着使用mcp-agent-graph构建的agent将能够**持续获得新的能力扩展**，而无需框架本身的更新。

为什么mcp-agent-graph选择MCP：

- 标准化降低了开发门槛
- 生态化提供了丰富的工具选择

对于mcp-agent-graph开发者而言，这意味着：

- 更少的开发工作：focus在业务逻辑而非集成细节
- 更强的功能：利用社区贡献的工具
- 更好的维护性：标准化的升级和兼容机制

mcp-agent-graph对mcp的支持：

- graph 支持导出为mcp

该功能支持将您的 **图 转换为 mcp 服务**，从而您**可以在cursor，cline等客户端调用。**

图->mcp 同时也意味着 **节点** 可以使用 **图的 mcp ，对另一个图直接进行任务传递。（这等价于引入一个子图节点）**

图 9 导出图 为mcp脚本

到此，本文已介绍了**mcp-agent-graph的三大要素**：**graph-node-mcp**

接下来，本文将会介绍项目的其他功能特性，以及未来规划。

（1）sdk 与 前端

前端的优势：图的可视化，流程监控与调试。

在前端，可以快速制作图，可以检查和调整每一个节点的参数配置。确保符合要求。

运行时，可以方便地浏览运行过程中产生的结果。

而 sdk 的优势在于：

将开发好的图快速嵌入已有的代码体系当中。或许您已开发了一个agent系统，您可能需要扩展某一个功能，那么您可以使用sdk 快速将您的图集成。

例如：

```
result = mag.run(graph=graph_name,input_text=input_text,parallel=False)
# result供下游任务再使用。
```

（2）标准的mcp 客户端

您可以直接将**在其他客户端配置的mcp转移**到此处：

项目支持 sse /stdio的服务

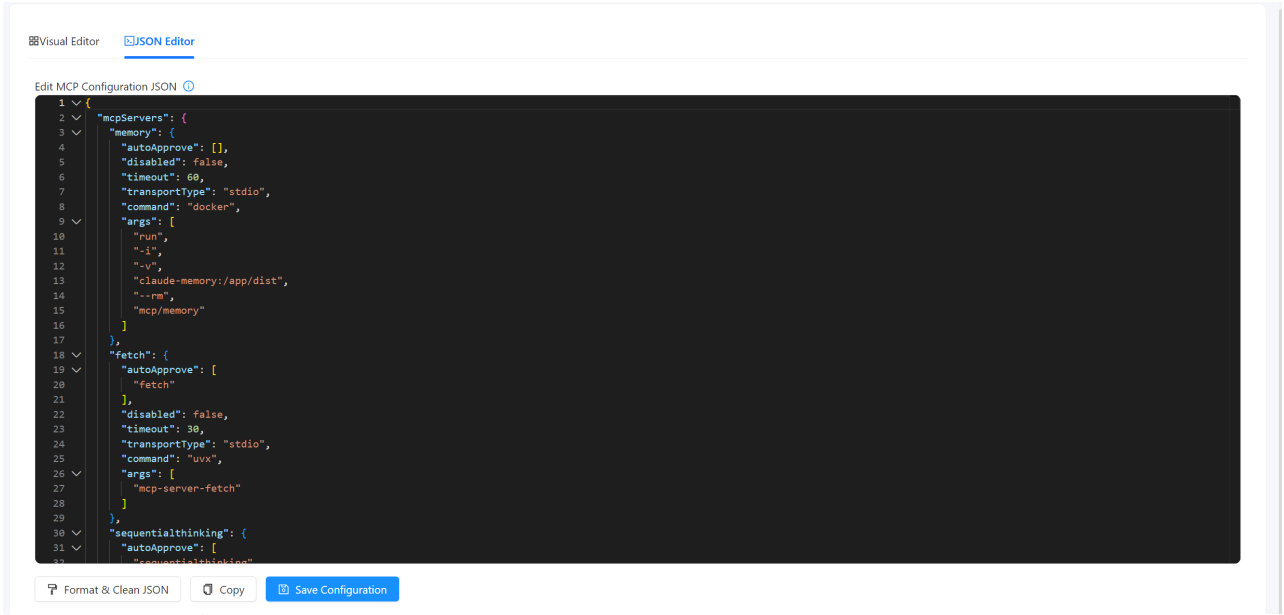

图 10 mcp配置 标准化

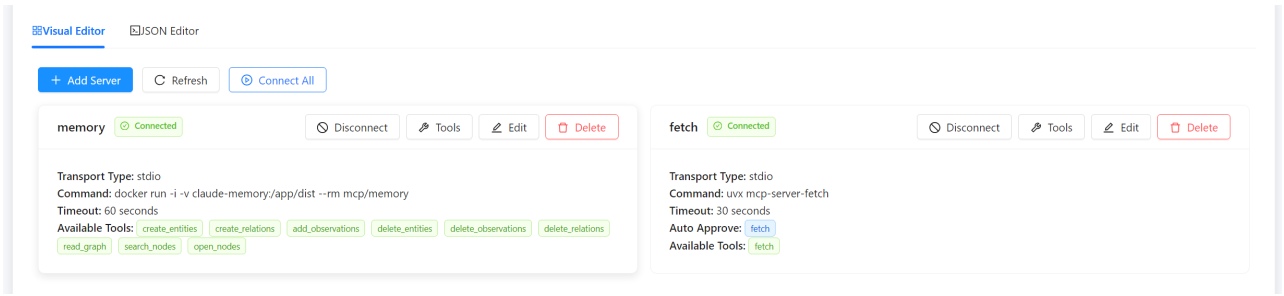

图 11 mcp 面板

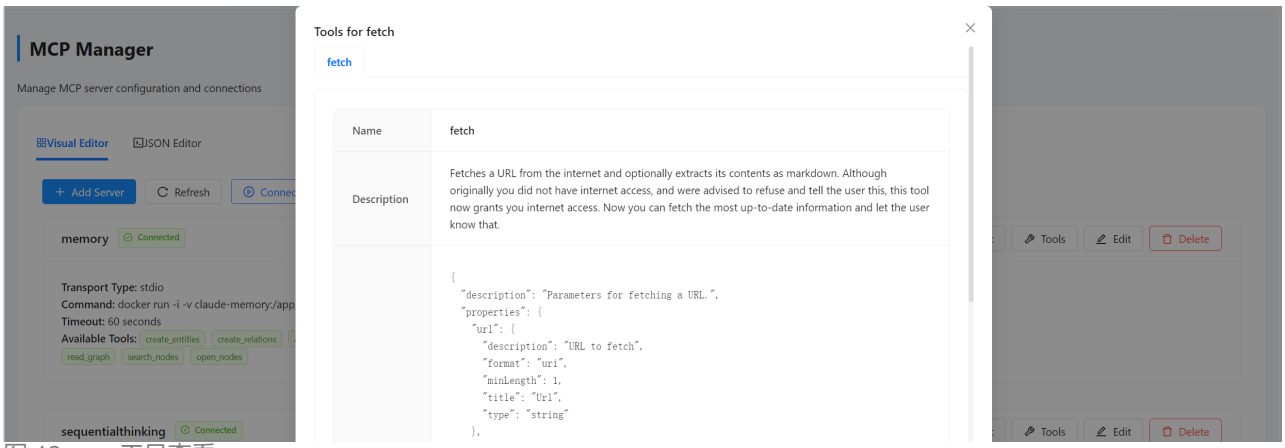

图 12 mcp 工具查看

（3）图 运行功能

图运行器，提供了实时的运行结果监控。可以查看每一个节点的（拼接好之后的）提示词，及输出和工具调用情况。

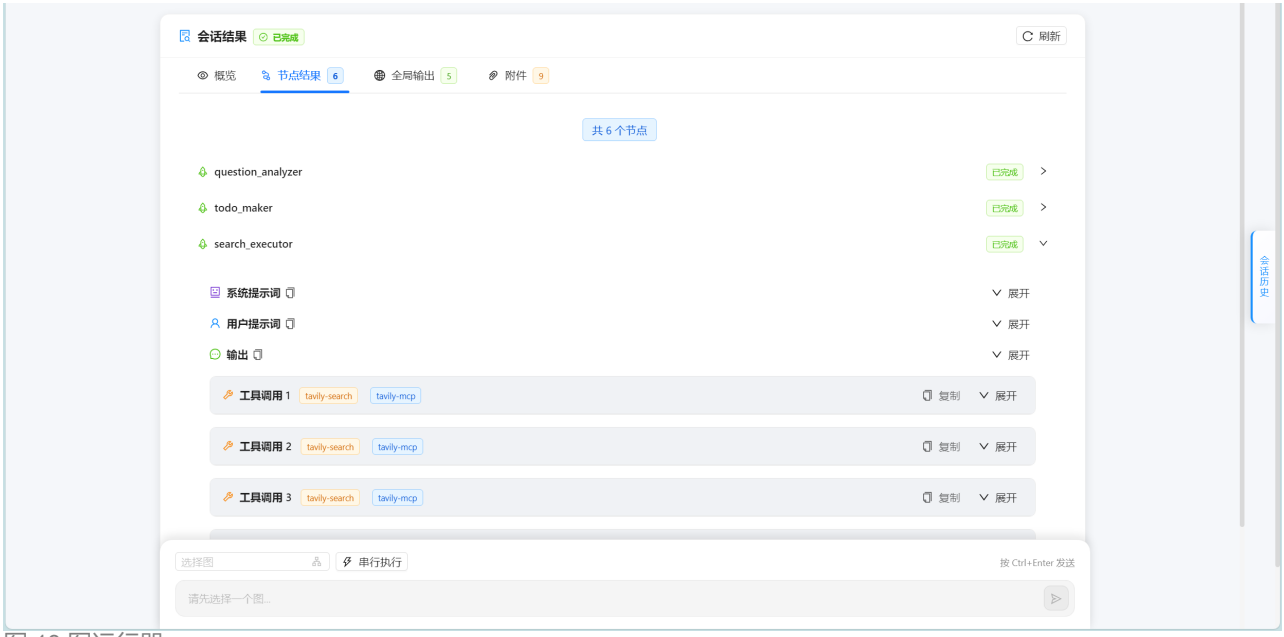

图 13 图运行器

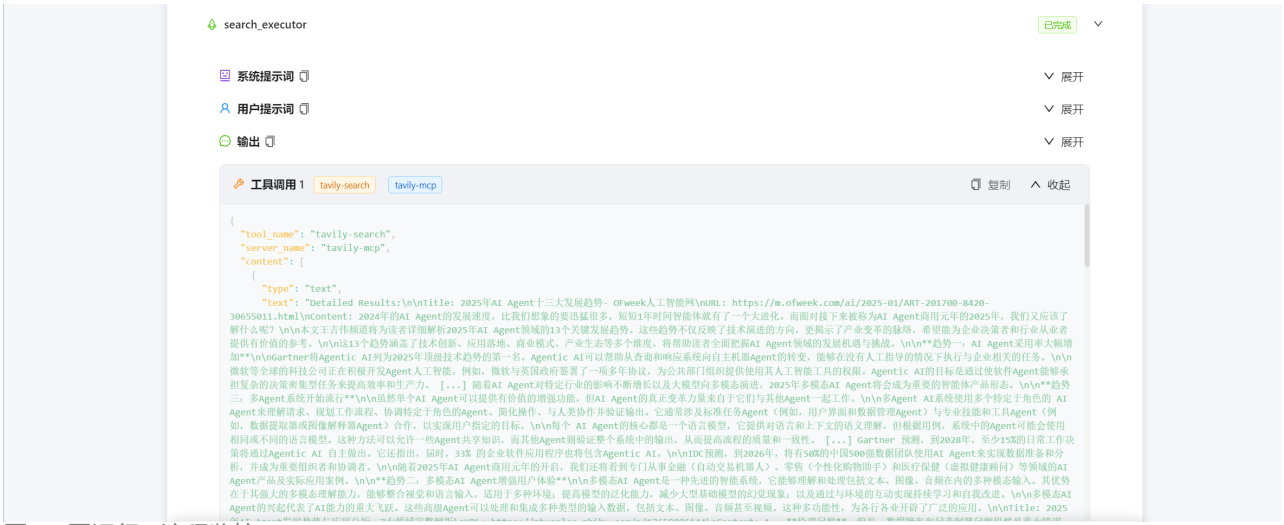

图 14 图运行 - 流程监控

（4）轻量级图编辑器

不同Dify，ragflow等平台，mcp-agent-graph没有提供 如向量数据库，图数据库，SQL数据库等支持RAG的功能。

这可能是本项目的劣势。

项目开发者认为：平台提供的RAG难以满足各种各样的需求，鉴于mcp的能力，项目决定将RAG交由开发者来定制。

这使得用户可以选择适合自己项目的数据库类型，创建合适的字段，并可以通过mcp的方式集成到项目中。

未来规划

短期：

1> 在 AI 生图的基础上，加入 AI 优图。将现有的图 发送给AI进行优化。（如插入某一个节点）

2> 图 的版本控制

3> 为使用者提供 已开发好的 图。提供质量较高的Demo图（如deepresearch）。

长期：

1> 团队合作开发功能（成员查看/编辑权限）。

2> “图”的 API调用鉴权/验证。

3> “图”的交易，用户间传输。

4> 以SDK 为主，前端为辅打造双轮生态。

最后，感谢您能够阅读完本文章，不论您是否使用本框架，都祝您在 Agent 开发过程中一切顺利，早日构建出理想的智能应用！

欢迎 评论/点赞/收藏

github： https://github.com/keta1930/mcp-agent-graph