零样本图推理通过检索增强框架与大语言模型实现

Hanqing Li[1], Kiran Sheena Jyothi[2],
Henry Liang[3], Sharika Mahadevan[4], Diego Klabjan[1]

[1]Northwestern University, [2]EXL Service, [3]Vail Systems, [4]Netflix

Correspondence: hanqingli2025@u.northwestern.edu

摘要

我们提出了一种新的、无需训练的方法——通过检索增强框架进行图推理（GRRAF），该方法利用检索增强生成（RAG）以及大型语言模型（LLMs）的代码生成能力来解决广泛的图推理任务。在 GRRAF 中，目标图存储在一个图数据库中，LLM 被提示生成可执行代码查询以检索必要的信息。这种方法避开了现有方法需要大量微调或依赖预定义算法的限制，并且它结合了一个带有超时机制的错误反馈循环，确保正确性和效率。在 GraphInstruct 数据集上的实验评估显示，GRRAF 在大多数图推理任务中实现了 100%的准确性，包括环检测、二分图检查、最短路径计算和最大流，同时保持了与图大小无关的一致代币成本。在子图匹配上也观察到了虽然不完美但仍然非常高的性能。值得注意的是，GRRAF 能够有效地扩展到具有多达 10,000 个节点的大型图。

1 介绍

图推理在跨多个领域的复杂系统建模和理解中发挥着关键作用 (Wu et al., 2020)。图形自然地表示实体及其在社交网络、运输系统、生物网络和通信基础设施等领域中的相互关系。确定连通性、检测循环和寻找最短路径等图推理任务不仅是理论计算机科学的核心，而且在网络优化、异常检测、决策支持系统等方面也有实际应用 (Scarselli et al., 2008)。然而，解决这些任务需要对图形拓扑有深入的理解，并结合精确的计算过程，强调了在当代机器学习研究中开发高效图推理方法的关键挑战 (Zhao et al., 2024)。

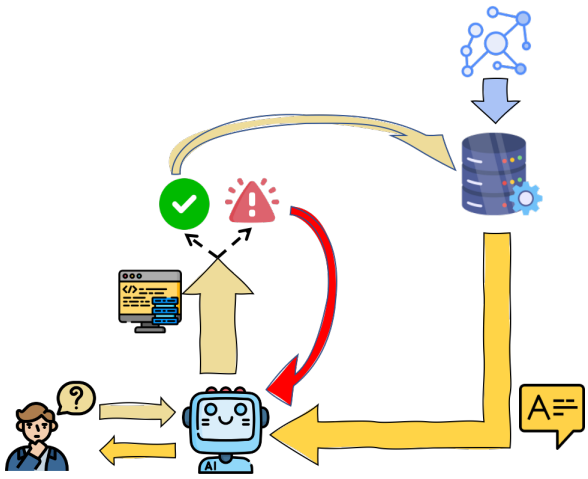

图 1: GRRAF 概念的示意图。当用户提出一个图形推理问题时，大语言模型会生成代码来查询存储在图形数据库中的目标图，检索答案，并将其作为响应呈现。错误反馈循环被集成到 GRRAF 中，以提示大语言模型在执行或超时时发生错误时优化代码。

大型语言模型（LLMs）展示了多步推理的惊人能力，这使它们能够解释用自然语言表达的复杂图相关问题，并生成可读性强的回应 (Guo et al., 2023)。最近的一些研究利用了 LLM 通过图神经网络（GNNs）将图形结构转换为文本表示或潜在嵌入的能力，从而利用了 LLMs 强大的自然语言推理能力 (Perozzi et al., 2024; Guo et al., 2023; Zhang, 2023; Wang et al., 2024a; Fatemi et al., 2024; Skianis et al., 2024; Lin et al.,

2024)。然而，即使使用先进的提示技术，这些方法在基础的图推理任务上表现仍然不佳，如评估连通性或识别最短路径，平均准确率范围从 20%到 60%。实现更高精度的替代方法通常要么需要大量的微调——这导致了对域外问题的性能较差 (Chen et al., 2024; Zhang, 2023)——或者依赖预定义算法作为输入，从而限制了它们处理未见过任务的能力 (Hu et al., 2024)。

为了解决这些限制，我们引入了一种无需训练且零样本的方法，**图推理通过检索增强框架**（GRRAF），该方法利用检索增强生成（RAG）(Lewis et al., 2020) 以及大型语言模型的代码编写能力。在 GRRAF 中，目标图存储在一个图形数据库中，而 LLM 被提示生成适当的查询，以代码形式书写这些查询，并通过从数据库中检索相关信息来提取所需的答案。这种策略利用了 LLM 强大的推理能力和其生成可执行代码的专业性，从而在一系列图形推理任务上实现了高精度，而不需额外的微调或预定义算法。此外，我们还结合了一个错误反馈回路和一个超时机制，以确保 LLM 能够高效地产生正确的查询。另外，由于准确的代码可靠地产生了正确答案，无论图的大小如何，GRRAF 都可以轻松扩展到多项式问题中处理更大的图形而不降低精度。在 GRRAF 中，我们使用了交互式的图形数据库 Neo4j 和用于图形的 Python 库 NetworkX。GRRAF 接受目标图为纯文本数据或已经存储在 Neo4j 中的数据，

GRRAF 提供了一个完全自动化的、端到端的框架，用于处理全部以文本形式书写的图推理问题。通过利用编码在大语言模型中的世界知识，它能够自动生成正确的代码并为一系列用自然语言表达的问题提供准确的答案。此外，GRRAF 为未来关于现实世界的结构化关系推断问题——从知识图谱补全到分子分析——建立了基础，这些问题自然而然地以图结构数据表示。

实验结果表明，GRRAF 在许多图推理任务上达到了 100%的准确率，超过了最先进的基准。此外，GRRAF 适用于包含多达 10,000 个节点的大图，在不增加代币成本的情况下保持 100%的准确率。尽管 GRRAF 在子图匹配上的准确率仅为 86.5%，但它仍然优于其他最先进的方法。我们的贡献如下列出。

- **新型图推理方法：**本研究介绍了一种利用 RAG 来解决图推理任务的新方法，包括连通性分析、环检测和最短路径计算。它代表了 RAG 在图推理领域中的首次应用。
- **误差反馈环创新：**本文介绍了在错误反馈回路中集成超时机制，并动态刷新提示以指导 LLM 生成更高效的代码。该机制通过防止无限循环来增强生成查询的鲁棒性和效率。
- **可扩展的前沿性能：**所提出的方法达到了最先进的准确性，并展示了卓越的可扩展性，成为第一个有效处理大型图且不会在准确性和成本方面显著下降的方法。

所有实现和数据集均可在 https://github.com/hanklee97121/GRRAF/tree/main 中获得。

2 相关工作

2.1 图 RAG

存在大量先前的研究，在 RAG 框架中使用图数据来增强 LLM 的能力，这种范式通常被称为 GraphRAG(Peng et al., 2024)。这些方法从预构建的图数据库 (Edge et al., 2024) 中检索包含与给定查询相关的关联知识的图元素。多项研究为 GraphRAG(Auer et al., 2007; Suchanek et al., 2007; Vrandečić and Krötzsch, 2014; Sap et al., 2019; Liu and Singh, 2004; Bollacker et al., 2008) 的开源知识图数据集的发展做出了贡献。基于这些数据集，许多方法选择将图转换为其他易于检索的形式，如文本 (Li et al., 2023; Huang et al., 2023; Yu et al., 2023; Edge et al., 2024; Dehghan et al., 2024) 或向量 (He et al., 2024; Sarmah et al., 2024)，以提高图

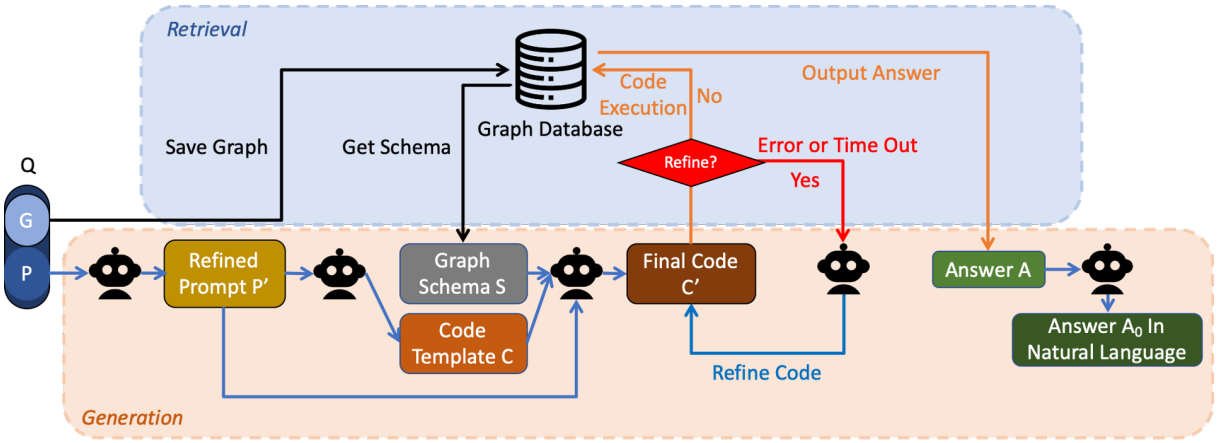

图 2: GRRAF 工作流。检索组件代表通过代码与图数据库的交互，而生成组件涉及提示 LLM 以产生输出。

数据库查询操作的效率。为进一步提升检索数据的质量，一些方法通过优化 GraphRAG 中的检索组件 (Delile et al., 2024; Zhang et al., 2022a; Kim et al., 2023; Wold et al., 2023; Jiang et al., 2023; Mavromatis and Karypis, 2024)、改进检索范式 (Wang et al., 2024b; Sun et al., 2024c; Lin et al., 2019) 和编辑用户查询或检索到的信息 (Jin et al., 2024; LUO et al., 2024; Ma et al., 2025; Sun et al., 2024a; Taunk et al., 2023; Yasunaga et al., 2021) 来完善检索过程。此外，许多方法增强了 GraphRAG 的答案生成过程，以确保大语言模型充分利用检索到的图数据来生成正确答案 (Dong et al., 2023; Mavromatis and Karypis, 2022; Jiang et al., 2024; Sun et al., 2024b; Zhang et al., 2022b; Zhu et al., 2024; Wen et al., 2024; Shu et al., 2022; Baek et al., 2023)。然而，这些方法仅专注于知识图谱，并不能直接应用于解决纯图形推理问题。相比之下，GRRAF 是第一个使用 RAG 来解决纯图形上的图形推理问题的方法。

2.2 图推理

最近的研究探讨了使用大语言模型解决图推理问题的可能性。几种方法完全依赖于提示工程技巧来增强大语言模型在图上的推理能力 (Liu and Wu, 2023; Guo et al., 2023; Wang et al., 2024a; Zhang et al., 2024; Fatemi et al., 2024; Wu et al., 2024; Tang et al., 2025; Skianis et al., 2024; Lin et al., 2024)。在此基础上，Perozzi et al. (2024) 将一个训练好的图神经网络 (Scarselli et al., 2008) 与大语言模型集成起来，通过将每个图编码成输入给大语言模型的标记来提高其在图推理任务上的性能。同时，Zhang (2023) 和 Chen et al. (2024) 对大语言模型进行微调，使用针对图推理任务定制的指令以提升性能。另一种方法中，Hu et al. (2024) 提出了一种多智能体解决方案来解决图推理问题，通过为每个节点分配一个大语言模型智能体，并基于预定义算法使智能体之间进行通信。相比之下，GRRAF 使用 RAG 来处理图推理问题，而无需大量的提示工程。这种方法是无训练的，因此是非监督的，并且不依赖于任何预定义算法。此外，与以前的方法不同，GRRAF 中的大语言模型不会接收整个图作为输入；因此，标记使用独立于图的大小，从而能够高效地扩展到非常大的图。

3 方法

在本节中，我们解释了 GRRAF 如何整合 RAG 来解决图推理问题并检索准确答案。整个 GRRAF 的工作流程如图 2 所示。一个图推理问题，记为 Q，包括两个组成部分：一个图 G 和一个用户提示 P。图 G 表示与 Q 关联的目标图，并存储为[redacted]提示 P 包含关于 G 的特定于图形的问题（例如，“节点 2 是否连接到节点 5?” 或 “从节点 5 到节点 8 的最短路径是什

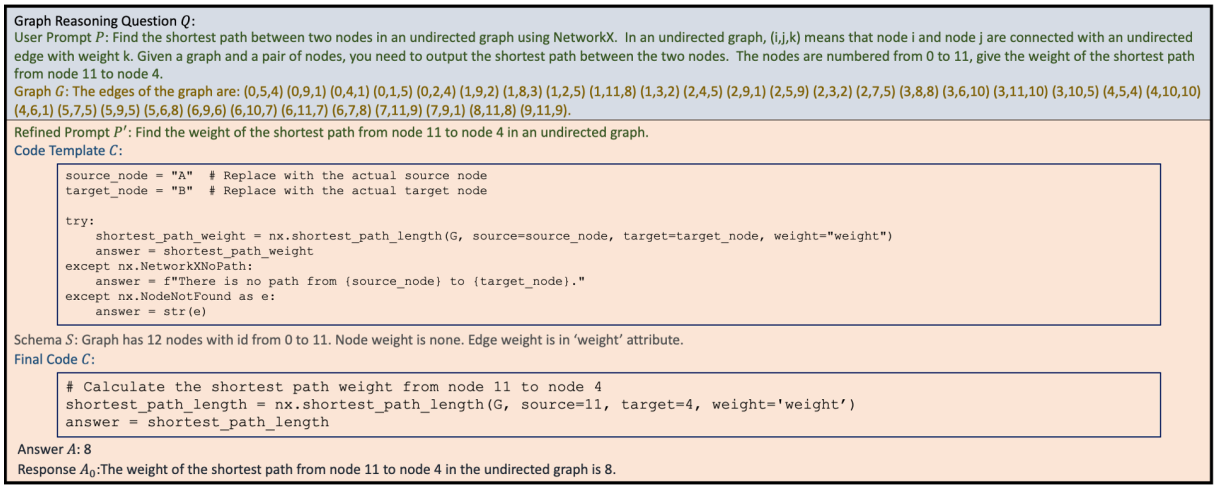

图 3: 一个演示将 GRRAF 应用于使用 NetworkX 解决最短路径问题的示例。文本中的图 G 通过代码存储为一个 NetworkX 对象。

任务	■大小	■图 的数量
Cycle Detection	[2, 100]	400
Connectivity	[2, 100]	400
Bipartite Graph Check	[2, 100]	400
Topological Sort	[2, 50]	400
Shortest Path	[2, 100]	400
Maximum Triangle Sum	[2, 25]	400
Maximum Flow	[2, 50]	400
Subgraph Matching	[2, 30]	400
Indegree Calculation	[2, 50]	400
Outdegree Calculation	[2, 50]	400

表 1: GraphInstruct 数据集的详细信息以及两个额外任务（入度计算和出度计算）。子图匹配任务是验证是否存在一个子图在 G 中与给定的图 G' 同构。

么?）。为了增强语言模型的代码生成，我们最初将 P 输入 LLM，请求它优化提示、澄清格式并消除冗余信息。经过优化的提示表示为 P'。然后，LLM 被提示生成一个通用代码模板 C 来解决 P' 问题而不包含特定于图的细节。例如，如果 P' 声明 "找到从节点 3 到节点 5 的最短路径"，那么模板 C 将封装一个不包括具体节点标识符的通用最短路径算法。此外，我们使用硬编码过程从图数据库中提取模式 S（包含节点属性和边属性）。此模式确保 LLM 生成的代码使用正确的变量名称。

随后，我们将 P'、C 和 S 提供给 LLM，并指示它生成最终代码 C'，该代码产生一个答案 A，与 P' 相对应。在此过程中加入了错误反馈循环。如果在执行 C' 过程中出现错误，会将错误消息连同 C' 一并反馈给大语言模型，促使其生成代码的修订版本。为了促进高效时间代码的生成，鉴于可能存在多种具有不同时间复杂度的算法适用情况，我们在错误反馈循环中集成了一个超时机制。具体来说，对 C' 的执行设定了一个时间限制 t。如果执行时间超过 t，则会终止该过程，并要求大语言模型修改 C' 以使其运行得更快。如果反馈循环迭代次数超过 n 次，系统将回退到使用原始问题 Q 作为提示，直接从 LLM 获取答案 A。这种强制退出设计旨在防止在处理计算上不可行的 NP 难问题（例如大型图中的子结构匹配）时出现无限迭代的情况，在这些问题中，对 C' 的任何修改都无法将执行时间降低到阈值 t 以下。

在最后一步，答案 A 提供给大语言模型以生成一个易于读者理解的自然语言响应 A_0，该响应解答了图推理问题 Q。使用 GRRAF 解决图推理问题的一个示例展示在图 3 中。

4 计算评估

4.1 数据集和基准测试

我们在 GraphInstruct(Chen et al., 2024) 数据集上进行了实验，该数据集包含九个不同复杂度的图形推理任务。由于其在图形推理任务上的多样性及其之前用于评估最先进的方法 (Chen et al., 2024; Hu et al., 2024) 的使用情况，我们选择了这个数据集进行评估。然而，寻找哈密尔顿路径的任务缺乏公开可用的真实标签，并且由于该问题的 NP 难性质，通过代码生成此类标签是不可行的；因此，我们将这项任务排除在实验之外。相应地，我们对 GRRAF 在以下八项任务上的性能进行了评估：环检测、连通性检查、二分图验证、拓扑排序、最短路径查找、最大三角形求和、最大流计算以及子图匹配。这些任务的详细信息见表 1。此外，为了实现更稳健的性能评估，我们在测试数据集中增加了两个简单的附加任务——入度计算和出度计算（如表 1 所示）——以促进对 GRRAF 和现有最先进技术基准的全面评估。每个任务包含 400 个问题-图对，每个都有一个正确答案。我们通过准确率衡量一种方法在一个任务上的表现，即正确回答的问题数占总问题数（400）的比例。

GRRAF，即其 LLM，生成的代码要么正确要么不正确。这就是为什么大多数[redacted]都将为 100%的原因。对于准确率低于 100%的任务，GRRAF 生成了正确的代码，但底层问题是 NP 难问题，并且对某些测试图来说执行时间超时。有人可能会认为输出的代码是正确的，因此应该给予适当的信用，但从另一方面来看，可以潜在地产生更有效的算法和代码。有时生成的代码不能正确处理边界情况，而其他时候代码或算法不正确（它们仅通过巧合解决了某些测试图）。

我们将 GRRAF 的性能与两个最先进的基准进行了比较：GraphWiz(Chen et al., 2024) 和 GAR(Hu et al., 2024)。GraphWiz 是在包含推理路径的 GraphInstruct 训练集上的 17,158 个问题和 72,785 个答案上进行训练的。由于没有单一版本的 GraphWiz 能够在所有任务中始终优于其他版本，我们在比较中包括了三个版本：GraphWiz (Mistral-7B)、GraphWiz-DPO (LLaMA 2-7B) 和 GraphWiz-DPO (LLaMA 2-13B)。GAR 是一个无需训练的多代理框架，依赖于由人类创建的预定义分布式算法库。因此，它无法解决其库中不存在算法的新颖图推理任务。因此，在随后的比较中，由于其局限性，一些来自 GAR 的结果缺失。

4.2 实验

我们使用 GRRAF 进行了实验，时间限制为 $t = 5$ 分钟，最大误差反馈循环迭代次数为 $n = 3$。骨干 LLM 是 GPT-4o。这些参数选择通过附录 4.3 中的敏感性分析得到验证。对于图查询代码，我们评估了两种方法：用于 Neo4j 的查询语言 Cypher 以及 Python 图库 NetworkX，并分别将其标记为 GRRAF$_C$ 和 GRRAF$_N$。[redacted]

图 4 表明 GARRF$_N$ 在大多数图形推理任务上超越了所有基准方法，达到了 100%的准确率。GARRF$_C$ 在大多数任务上的表现与其它基准相当或更优，除了拓扑排序和子图匹配。尽管 GraphWiz 在拓扑排序和子图匹配方面优于 GARRF$_C$，但在入度计算和出度计算上表现不佳，表明它难以处理即使是简单的跨域图形推理任务。此外，由于其固有限制，GAR 不适用于最大流、子图匹配、入度计算和出度计算等跨域任务。因此，考虑到性能和泛化能力，GARRF$_C$ 和 GARRF$_N$ 比其他基准模型更适合解决图形推理任务。由 GARRF$_N$ 生成的每个图形推理任务的示例代码见附录 A。

[redacted]基于第 3 节，在这种情况下 GARRD$_N$ 回退到使用原始问题 Q 作为提示以直接从大语言模型获取答案 A，这可能会产生不正确的结果。GRRAF$_C$ 在环检测和二分图检查上的准确率也未能达到 100%，因为 Cypher 查询的执行速度比 NetworkX 慢。对于最大流任务，GRRAF$_C$ 生成的代码忽略了某些边界情况。而对于拓扑排序和子图匹配，它生成的代码仅在

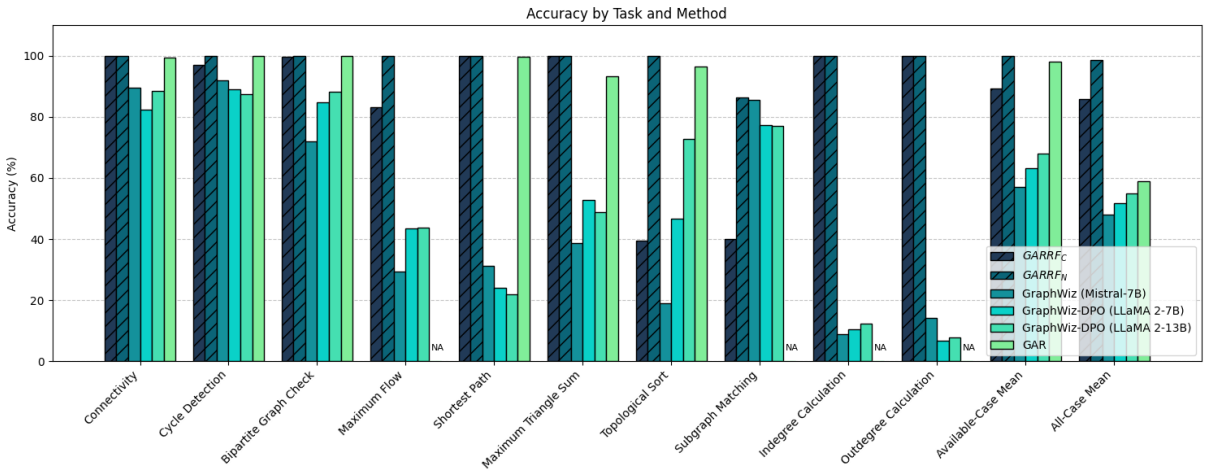

图 4: GRRAF 和基准模型在所有十个图形推理任务中的性能。图中用 “NA” 表示缺失数据。可获得情况下的平均值是指使用仅包含完整数据的任务计算的每种方法的平均准确率（排除最大流、子图匹配、入度计算和出度计算）。全部情况下的平均值是指将 ’NA’ 视为 0 的所有任务上的平均准确率。

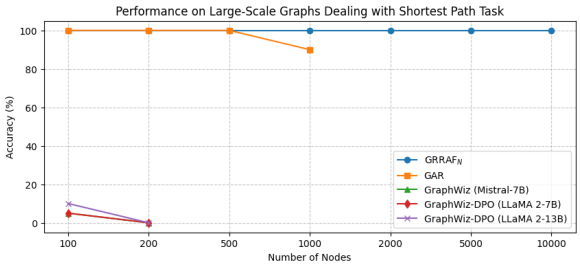

图 5: 每种方法在不同规模（节点数量）的图上最短路径任务的准确性。

某些图上偶然成功。

在十个任务中，解答一个单一的图推理问题需要 GRRAF$_N$ 平均使用 767 个输入标记和 124 个输出标记，而 GRRAF$_C$ 则使用了 796 个输入标记和 201 个输出标记。相比之下，GraphWiz（Mistral-7B）每个问题平均消耗 1,046 个输入标记和 126 个输出标记，而 GraphWiz-DPO（LLaMA 2-7B）则需要平均 1,046 个输入标记和 290 个输出标记，GraphWiz-DPO（LLaMA 2-13B）每个问题使用了 1,046 个输入标记和 301 个输出标记。值得注意的是，GAR 需要更多的资源，每解决一个图推理问题平均需 8,095 个输入标记和 5,987 个输出标记。因此，与其他基准方法相比，GRRAF$_N$ 和 GRRAF$_C$ 在利用较少的标记资源的同时，在图推理任务中实现了高精度。

由于 GraphInstruct 中的最大图 (Chen et al., 2024) 只包含 100 个节点，这对于实际的图形推理场景 (Hu et al., 2024) 来说是远远不够的，因此我们进一步评估了表现最佳的方法 GRRAF$_N$ 在大规模图形上的性能。遵循 Hu et al. (2024) 的方法，我们在更大的图形上使用 20 个测试样本对每个图形大小进行评估 GRRAF$_N$ 的最短路径任务。虽然他们的工作将图形扩展到 1,000 个节点，但我们通过将图形扩展到 10,000 个节点来进一步评估 GRRAF$_N$ 的性能。根据图 5，GRRAF$_N$ 在所有图形大小上均实现了 100%的准确率，展示了其出色的可扩展性。GAR 在包含 100、200 和 500 个节点的图上达到了 100%的准确率，但在包含 1,000 个节点的图上的准确率下降到 90%。由于令牌限制，GAR 无法处理包含 2,000 个或更多节点的问题。相比之下，GraphWiz 的所有三个版本在大型图形上表现不佳，在包含 100 个节点的图形上仅达到 5-10%的准确率，并且完全未能处理包含 200 个节点的图形。它们基础模型的令牌限制阻止了它们处理超过 200 个节点的图形。

我们还记录了解决单个图推理问题所需的 token 成本随着图大小在最短路径任务中增加的变化。

如图 6 所示，GRRAF$_N$ 使用的 token 数量

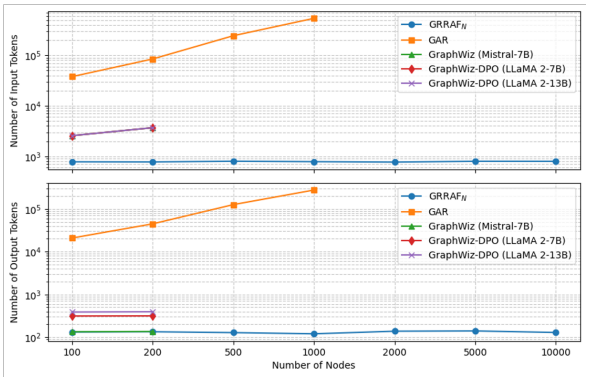

图 6: 解决不同大小图上的最短路径任务的图形推理问题的平均令牌成本。

方法	执行错误	超时
GRRAF$_N$	2.2%	5.4%
GRRAF$_C$	4.9%	9.1%

表 2: 各方法因执行错误或超时而在 10 项任务中触发错误反馈循环的图形推理问题的比例。

与图的大小无关，保持不变。

如 3 节所述，GRRAF 仅通过代码执行与图数据库交互；因此，图的描述（节点、边、权重）不会直接输入到 LLM 中，并且 token 成本不受图大小增加的影响。

相比之下，GraphWiz 的 token 成本随着图大小线性增加，因为它必须将每个节点和边的信息传递给 LLM。

GAR 的 token 成本远高于 GRRAF$_N$，并且随着图大小几乎呈指数级增长。

这是由于 GAR 的设计，其中每个节点都被分配一个 LLM 代理，并且每个代理在每次迭代 (Hu et al., 2024) 中与每个相邻代理通信。

随着节点数量的增加，代理数量、每个节点的相邻代理数量（即边）以及获得答案所需的迭代次数都会增加，所有这些都导致 token 成本显著上升。

因此，与其他基准相比，GRRAF$_N$ 可以轻松扩展到非常大的图（最多 10,000 个节点），而不会影响性能并增加 token 成本。

为了评估错误反馈环的有效性，我们量化了激活此环的问题的总百分比，如表 2 所示。总体而言，GRRAF$_C$ 比 GRRAF$_N$ 更频繁地触发错误反馈环。对于这两种变体，该环由于超时而被激活的频率高于因执行错误而被激活的频率，强调了时间效率在图推理任务中的重要性。总体而言，骨干大语言模型在大多数情况下生成正确的代码查询，并且与超时机制结合的错误反馈环进一步提高了代码的准确性和效率。

4.3 敏感性分析

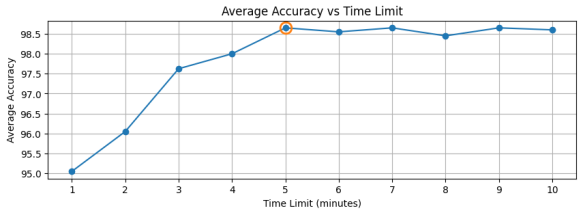

图 7: GRRAF$_N$ 在不同时间限制 t 下的平均准确率。

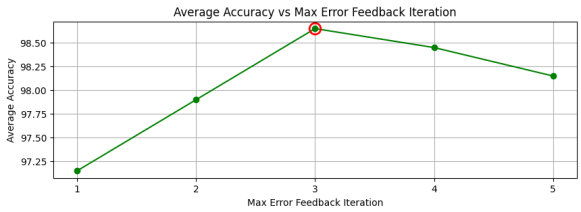

图 8: GRRAF$_N$ 的平均准确率与不同的最大误差反馈循环迭代次数 n。

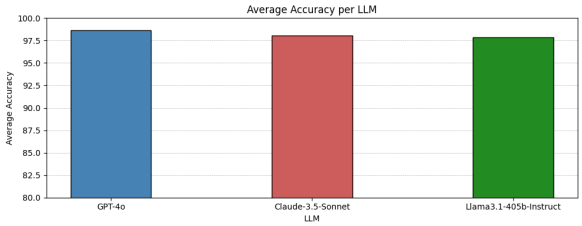

图 9: GRRAF$_N$ 使用不同主干 LLM 的平均准确率。

我们对 GRRAF$_N$ 进行了敏感性分析，以评估时间限制 t、最大错误反馈循环迭代次数 n 和选择骨干大语言模型的影响。我们报告了所有十个图推理任务的平均准确率。如图 7 所示，准确率随 t 增加至五分钟，之后不再有进一步提高。图 8 表明准确性在 $n = 3$ 处达到

峰值，并且对于 $n > 3$ 略有下降。最后，我们使用三个基础大语言模型——GPT-4o、Claude-3.5-Sonnet 和 Llama3.1-405b-Instruct——评估了 GRRAF$_N$，发现所有三种模型的结果相当接近，其中 GPT-4o 达到了略高于其他两种模型的平均准确性（图 9）。

5 结论

在这项工作中，我们介绍了 GRRAF，一个将 RAG 与 LLMs 的代码编写能力相结合的新框架，用于解决图推理问题。我们的方法无需额外训练或依赖预定义算法，利用图数据库存储目标图，并使用带有超时机制的错误反馈循环来确保生成正确且高效的代码查询。在 GraphInstruct 数据集和两个额外任务（入度和出度）上的全面实验表明，GRRAF 优于现有的最先进的基准，在大多数图推理任务上实现了 100%的准确性，同时能够有效地扩展到包含多达 10,000 个节点的图形，而不会产生额外的令牌成本。这些发现强调了将基于检索的技术与 LLM 驱动的代码生成相结合解决复杂图推理问题的潜力。未来的工作可以探索将此框架扩展到动态图场景和更多推理任务，进一步增强其适用性和鲁棒性。

6 限制

尽管 GRRAF$_N$ 在所有多项式时间图推理任务上达到了 100% 的准确率，但它仍然难以在精确和高效地解决 NP 完全问题——例如子图匹配。此外，GRRAF$_C$ 相对于 GRRAF$_N$ 表现较差表明我们的框架目前生成的 Cypher 查询质量低于等效的 Python 代码。这两个问题是我们的方法的主要限制。

References

Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. 2007. Dbpedia: A nucleus for a web of open data. In *International Semantic Web Conference*, pages 722–735.

Jinheon Baek, Soyeong Jeong, Minki Kang, Jong Park, and Sung Hwang. 2023. Knowledge-augmented language model verification. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 1720–1736.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, pages 1247–1250.

Nuo Chen, Yuhan Li, Jianheng Tang, and Jia Li. 2024. Graphwiz: An instruction-following language model for graph computational problems. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 353–364.

Mohammad Dehghan, Mohammad Alomrani, Sunyam Bagga, David Alfonso-Hermelo, Khalil Bibi, Abbas Ghaddar, Yingxue Zhang, Xiaoguang Li, Jianye Hao, Qun Liu, Jimmy Lin, Boxing Chen, Prasanna Parthasarathi, Mahdi Biparva, and Mehdi Rezagholizadeh. 2024. EWEK-QA : Enhanced web and efficient knowledge graph retrieval for citation-based question answering systems. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 14169–14187.

Julien Delile, Srayanta Mukherjee, Anton Van Pamel, and Leonid Zhukov. 2024. Graph-based retriever captures the long tail of biomedical knowledge. In *ICML'24 Workshop ML for Life and Material Science: From Theory to Industry Applications*.

Junnan Dong, Qinggang Zhang, Xiao Huang, Keyu Duan, Qiaoyu Tan, and Zhimeng Jiang. 2023. Hierarchy-aware multi-hop question answering over knowledge graphs. In *Proceedings of the ACM web conference 2023*, pages 2519–2527.

Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, and Jonathan Larson. 2024. From local to global: A graph RAG approach to query-focused summarization. *arXiv preprint arXiv:2404.16130*.

Bahare Fatemi, Jonathan Halcrow, and Bryan Perozzi. 2024. Talk like a graph: Encoding graphs for large language models. In *The Twelfth International Conference on Learning Representations*.

Jiayan Guo, Lun Du, Hengyu Liu, Mengyu Zhou, Xinyi He, and Shi Han. 2023. GPT4Graph: Can large language models understand graph structured data? an empirical evaluation and benchmarking. In *arXiv preprint arXiv:2305.15066*.

Xiaoxin He, Yijun Tian, Yifei Sun, Nitesh V Chawla, Thomas Laurent, Yann LeCun, Xavier Bresson, and Bryan Hooi. 2024. G-retriever: Retrieval-augmented generation for textual graph understanding and question answering. In *The Thirty-eighth*

Annual Conference on Neural Information Processing Systems.

Yuwei Hu, Runlin Lei, Xinyi Huang, Zhewei Wei, and Yongchao Liu. 2024. Scalable and accurate graph reasoning with LLM-based multi-agents. In *arXiv preprint arXiv:2410.05130*.

Yongfeng Huang, Yanyang Li, Yichong Xu, Lin Zhang, Ruyi Gan, Jiaxing Zhang, and Liwei Wang. 2023. MVP-Tuning: Multi-view knowledge retrieval with prompt tuning for commonsense reasoning. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13417–13432.

Boran Jiang, Yuqi Wang, Yi Luo, Dawei He, Peng Cheng, and Liangcai Gao. 2024. Reasoning on efficient knowledge paths: knowledge graph guides large language model for domain question answering. In *2024 IEEE International Conference on Knowledge Graph*, pages 142–149.

Jinhao Jiang, Kun Zhou, Zican Dong, Keming Ye, Xin Zhao, and Ji-Rong Wen. 2023. StructGPT: A general framework for large language model to reason over structured data. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 9237–9251.

Bowen Jin, Chulin Xie, Jiawei Zhang, Kashob Kumar Roy, Yu Zhang, Zheng Li, Ruirui Li, Xianfeng Tang, Suhang Wang, Yu Meng, and Jiawei Han. 2024. Graph chain-of-thought: Augmenting large language models by reasoning on graphs. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 163–184.

Jiho Kim, Yeonsu Kwon, Yohan Jo, and Edward Choi. 2023. KG-GPT: A general framework for reasoning on knowledge graphs using large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Advances in Neural Information Processing Systems*, volume 33, pages 9459–9474.

Shiyang Li, Yifan Gao, Haoming Jiang, Qingyu Yin, Zheng Li, Xifeng Yan, Chao Zhang, and Bing Yin. 2023. Graph reasoning for question answering with triplet retrieval. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 3366–3375.

Bill Yuchen Lin, Xinyue Chen, Jamin Chen, and Xiang Ren. 2019. KagNet: Knowledge-aware graph networks for commonsense reasoning. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 2829–2839.

Tianqianjin Lin, Pengwei Yan, Kaisong Song, Zhuoren Jiang, Yangyang Kang, Jun Lin, Weikang Yuan, Junjie Cao, Changlong Sun, and Xiaozhong Liu. 2024. LangGFM: A large language model alone can be a powerful graph foundation model. In *arXiv preprint arXiv:2410.14961*.

Chang Liu and Bo Wu. 2023. Evaluating large language models on graphs: Performance insights and comparative analysis. In *arXiv preprint arXiv:2308.11224*.

Hugo Liu and Push Singh. 2004. ConceptNet—a practical commonsense reasoning tool-kit. In *BT Technology Journal*, volume 22, pages 211–226.

LINHAO LUO, Yuan-Fang Li, Reza Haf, and Shirui Pan. 2024. Reasoning on graphs: Faithful and interpretable large language model reasoning. In *The Twelfth International Conference on Learning Representations*.

Shengjie Ma, Chengjin Xu, Xuhui Jiang, Muzhi Li, Huaren Qu, Cehao Yang, Jiaxin Mao, and Jian Guo. 2025. Think-on-graph 2.0: Deep and faithful large language model reasoning with knowledge-guided retrieval augmented generation. In *The Thirteenth International Conference on Learning Representations*.

Costas Mavromatis and George Karypis. 2022. ReaRev: Adaptive reasoning for question answering over knowledge graphs. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 2447–2458.

Costas Mavromatis and George Karypis. 2024. GNN-RAG: Graph neural retrieval for large language model reasoning. In *arXiv preprint arXiv:2405.20139*.

Boci Peng, Yun Zhu, Yongchao Liu, Xiaohe Bo, Haizhou Shi, Chuntao Hong, Yan Zhang, and Siliang Tang. 2024. Graph retrieval-augmented generation: A survey. In *arXiv preprint arXiv:2408.08921*.

Bryan Perozzi, Bahare Fatemi, Dustin Zelle, Anton Tsitsulin, Mehran Kazemi, Rami Al-Rfou, and Jonathan Halcrow. 2024. Let your graph do the talking: Encoding structured data for LLMs. In *arXiv preprint arXiv:2402.05862*.

Maarten Sap, Ronan Le Bras, Emily Allaway, Chandra Bhagavatula, Nicholas Lourie, Hannah Rashkin, Brendan Roof, Noah A Smith, and Yejin Choi. 2019. Atomic: An atlas of machine commonsense for if-then reasoning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3027–3035.

Bhaskarjit Sarmah, Dhagash Mehta, Benika Hall, Rohan Rao, Sunil Patel, and Stefano Pasquali. 2024. HybridRAG: Integrating knowledge graphs and vector retrieval augmented generation for efficient information extraction. In *Proceedings of the 5th ACM International Conference on AI in Finance*, pages 608–616.

Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. 2008. The graph neural network model. In *IEEE Transactions on Neural Networks*, volume 20, pages 61–80.

Yiheng Shu, Zhiwei Yu, Yuhan Li, Börje Karlsson, Tingting Ma, Yuzhong Qu, and Chin-Yew Lin. 2022. TIARA: Multi-grained retrieval for robust question answering over large knowledge base. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 8108–8121.

Konstantinos Skianis, Giannis Nikolentzos, and Michalis Vazirgiannis. 2024. Graph reasoning with large language models via pseudo-code prompting. In *arXiv preprint arXiv:2409.17906*.

Fabian M Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. Yago: a core of semantic knowledge. In *Proceedings of the 16th International Conference on World Wide Web*, pages 697–706.

Jiashuo Sun, Chengjin Xu, Lumingyuan Tang, Saizhuo Wang, Chen Lin, Yeyun Gong, Lionel Ni, Heung-Yeung Shum, and Jian Guo. 2024a. Think-on-Graph: Deep and responsible reasoning of large language model on knowledge graph. In *The Twelfth International Conference on Learning Representations*.

Jiashuo Sun, Chengjin Xu, Lumingyuan Tang, Saizhuo Wang, Chen Lin, Yeyun Gong, Lionel Ni, Heung-Yeung Shum, and Jian Guo. 2024b. Think-on-graph: Deep and responsible reasoning of large language model on knowledge graph. In *The Twelfth International Conference on Learning Representations*.

Lei Sun, Zhengwei Tao, Youdi Li, and Hiroshi Arakawa. 2024c. ODA: Observation-driven agent for integrating LLMs and knowledge graphs. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 7417–7431.

Jianheng Tang, Qifan Zhang, Yuhan Li, Nuo Chen, and Jia Li. 2025. GraphArena: Evaluating and exploring large language models on graph computation. In *The Thirteenth International Conference on Learning Representations*.

Dhaval Taunk, Lakshya Khanna, Siri Venkata Pavan Kumar Kandru, Vasudeva Varma, Charu Sharma, and Makarand Tapaswi. 2023. GrapeQA: Graph augmentation and pruning to enhance question-answering. In *Companion Proceedings of the ACM Web Conference 2023*, pages 1138–1144.

Denny Vrandečić and Markus Krötzsch. 2014. Wikidata: a free collaborative knowledgebase. In *Communications of the ACM*, volume 57, pages 78–85.

Heng Wang, Shangbin Feng, Tianxing He, Zhaoxuan Tan, Xiaochuang Han, and Yulia Tsvetkov. 2024a. Can language models solve graph problems in natural language? In *Advances in Neural Information Processing Systems*, volume 36.

Yu Wang, Nedim Lipka, Ryan A Rossi, Alexa Siu, Ruiyi Zhang, and Tyler Derr. 2024b. Knowledge graph prompting for multi-document question answering. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 19206–19214.

Yilin Wen, Zifeng Wang, and Jimeng Sun. 2024. MindMap: Knowledge graph prompting sparks graph of thoughts in large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10370–10388.

Sondre Wold, Lilja Øvrelid, and Erik Velldal. 2023. Text-to-KG alignment: Comparing current methods on classification tasks. In *Proceedings of the First Workshop on Matching From Unstructured and Structured Data*, pages 1–13.

Qiming Wu, Zichen Chen, Will Corcoran, Misha Sra, and Ambuj K Singh. 2024. Grapheval2000: Benchmarking and improving large language models on graph datasets. In *arXiv preprint arXiv:2406.16176*.

Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. 2020. A comprehensive survey on graph neural networks. In *IEEE Transactions on Neural Networks and Learning Systems*, volume 32, pages 4–24.

Michihiro Yasunaga, Hongyu Ren, Antoine Bosselut, Percy Liang, and Jure Leskovec. 2021. QA-GNN: Reasoning with language models and knowledge graphs for question answering. In *North American Chapter of the Association for Computational Linguistics*.

Donghan Yu, Sheng Zhang, Patrick Ng, Henghui Zhu, Alexander Hanbo Li, Jun Wang, Yiqun Hu, William Yang Wang, Zhiguo Wang, and Bing Xiang. 2023. DecAF: Joint decoding of answers and logical forms for question answering over knowledge bases. In *The Eleventh International Conference on Learning Representations*.

Jiawei Zhang. 2023. Graph-toolformer: To empower LLMs with graph reasoning ability via prompt augmented by chatgpt. In *arXiv preprint arXiv:2304.11116*.

Jing Zhang, Xiaokang Zhang, Jifan Yu, Jian Tang, Jie Tang, Cuiping Li, and Hong Chen. 2022a. Subgraph retrieval enhanced model for multi-hop knowledge base question answering. In *Proceedings of the 60th Annual Meeting of the Association*

for Computational Linguistics (Volume 1: Long Papers), pages 5773–5784.

Xikun Zhang, Antoine Bosselut, Michihiro Yasunaga, Hongyu Ren, Percy Liang, Christopher D Manning, and Jure Leskovec. 2022b. GreaseLM: Graph REASoning enhanced language models. In *International Conference on Learning Representations*.

Zeyang Zhang, Xin Wang, Ziwei Zhang, Haoyang Li, Yijian Qin, and Wenwu Zhu. 2024. LLM4DyG: Can large language models solve spatial-temporal problems on dynamic graphs? In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, page 4350 – 4361.

Jianan Zhao, Le Zhuo, Yikang Shen, Meng Qu, Kai Liu, Michael M. Bronstein, Zhaocheng Zhu, and Jian Tang. 2024. Graphtext: Graph reasoning in text space. In *Adaptive Foundation Models: Evolving AI for Personalized and Efficient Learning*.

Yun Zhu, Yaoke Wang, Haizhou Shi, and Siliang Tang. 2024. Efficient tuning and inference for large language models on textual graphs. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence*, pages 5734–5742.

A 示例代码

本节展示了由 GRRAF$_N$ 生成的每个图推理任务的示例代码：环检测（图 10）、连通性（图 11）、二分图检查（图 12）、拓扑排序（图 13）、最短路径（图 14）、最大三角形和（图 15）、最大流（图 16）、子图匹配（图 17）、入度计算（图 18）以及出度计算（图 19）。■

我们在图 20 中包含了一个由 GRRAF$_C$ 生成的示例 Cypher 查询，用于■任务。尽管此查询试图实现 Ford – Fulkerson 算法，但它遗漏了反向残余边，阻止了早期流的重新路由。因此，在某些边缘情况下（例如图 21 中的图形），它会产生不正确的结果。类似地，图 22 显示了一个实例，其中 GRRAF$_C$ 为拓扑排序生成了错误的 Cypher 查询。该查询构建以零入度节点为根的生成树来推导顺序——这种方法是不合理的，并且只在某些图上偶然成功。

Cycle Detection

```
try: # Attempt to find a cycle in the graph G
     nx.find_cycle(G, orientation='ignore')
     answer = "Yes."
except nx.exception.NetworkXNoCycle:
     answer = "No."
```

图 10: 一个用于循环检测任务生成的最终代码示例 C'。

Connectivity

```
# Check if there is a path between node 8 and node 10 in the graph G
if nx.has_path(G, 8, 10):
     answer = "Yes."
else:
     answer = "No."
```

图 11: 连接任务生成的最终代码示例 C'。

Bipartite Graph Check

```
# Check if the directed graph G is bipartite
is_bipartite = nx.is_bipartite(G)
# Generate the answer string
answer = "Yes" if is_bipartite else "No"
```

图 12: 二分图检查任务生成的最终代码示例 C'。

Topological Sort

```
# Assuming G is already defined and is a directed graph
# Find one topological sorting path of the graph
topological_order = list(nx.topological_sort(G))
# Convert the list to a string representation
answer = ' -> '.join(map(str, topological_order))
```

图 13: 拓扑排序任务生成的最终代码示例 C'。

Shortest Path

```
# Calculate the shortest path weight from node 11 to node 4
shortest_path_length = nx.shortest_path_length(G, source=11, target=4, weight='weight')
answer = shortest_path_length
```

图 14: 最短路径任务生成的最终代码示例 C'。

Maximum Triangle Sum

```
from itertools import combinations
# Assume G is already defined as a directed graph
max_sum = -float('inf')
max_nodes = None
# Iterate over all combinations of three nodes in the graph
for nodes in combinations(G.nodes, 3):
     a, b, c = nodes
     # Check if the three nodes are interconnected, i.e., all pairs have edges in both directions
     if (G.has_edge(a, b) and G.has_edge(b, a) and G.has_edge(b, c) and G.has_edge(c, b) and G.has_edge(a, c) and
     G.has_edge(c, a)):
          # Calculate the sum of their weights
          total_weight = (G.nodes[a].get('weight', 0) + G.nodes[b].get('weight', 0) + G.nodes[c].get('weight', 0))
          # Update the maximum sum and the corresponding node set if needed
          if total_weight > max_sum:
          max_sum = total_weight
          max_nodes = nodes
# Generate the answer string with the maximum sum
answer = f"Answer is {max_sum}."
```

图 15: 最大三角形和任务生成的最终代码示例 C'。

Maximum Flow

```
# Calculate the maximum flow from node 0 to node 8
flow_value, flow_dict = nx.maximum_flow(G, 0, 8)
# Store the result in the variable 'answer'
answer = str(flow_value)
```

图 16: 最大流任务生成的最终代码示例 C'。

Subgraph Matching

```
from networkx.algorithms import isomorphism
import itertools
# First, try to find a subgraph isomorphic to G_prime using the standard DiGraphMatcher
matcher = isomorphism.DiGraphMatcher(G, G_prime)
if matcher.subgraph_is_isomorphic():
    # We found a subgraph isomorphic to G_prime (this covers the node-induced scenario)
    answer = "Yes, G_prime is a subgraph of G."
else:
    # If no node-induced subgraph match was found, we try edge-induced.
    # An edge-induced subgraph of G is defined by a subset of edges of G.
    # We must find a set of edges in G that, when taken alone (with their endpoints),
    # forms a subgraph isomorphic to G_prime.
    # Get edges and nodes of G_prime for reference
    G_prime_edges = list(G_prime.edges())
    G_prime_edge_count = len(G_prime_edges)
    G_prime_nodes = set(G_prime.nodes())
    found_edge_induced = False
    for edge_subset in itertools.combinations(G.edges(), G_prime_edge_count):
        # Build a subgraph of G induced by these edges
        sub_nodes = set()
        for u, v in edge_subset:
            sub_nodes.add(u)
            sub_nodes.add(v)
        # Create a new directed graph with just these edges
        sub_G = G.subgraph(sub_nodes).copy()
        # Remove all edges not in edge_subset
        sub_G.remove_edges_from([e for e in sub_G.edges() if e not in edge_subset])
        # Now check if sub_G is isomorphic to G_prime
        test_matcher = isomorphism.DiGraphMatcher(sub_G, G_prime)
        if test_matcher.is_isomorphic():
                found_edge_induced = True
                break
    if found_edge_induced:
        answer = "Yes, G_prime is a subgraph of G."
    else:
        answer = "No, G_prime is not a subgraph of G."
```

图 17: 子图匹配任务生成的最终代码示例 C'。

Indegree Calculation

```
# Assuming G is already defined as a directed graph in the existing code
indegree_node_5 = G.in_degree(5)
answer = f"{indegree_node_5}"
```

图 18: 一个用于计算入度任务生成的最终代码示例 C'。

Outdegree Calculation

```
# Calculate the outdegree of node 2
outdegree_node_2 = G.out_degree(2)
# Create a string with the result
answer = f"{outdegree_node_2}."
```

图 19: 一个用于出度计算任务生成的最终代码示例 C'。

Maximum Flow (Cypher)

```
" MATCH ()-[r:CONNECTS_flow_8]->() SET r.flow = 0 "

" // 1. Find any path from source(id=2) to sink(id=6)
//    where every edge still has spare capacity (weight > flow)
MATCH path = (source:Node_flow_8 {id: 2})-[:CONNECTS_flow_8*]->(sink:Node_flow_8 {id: 6})
WHERE ALL(r IN relationships(path) WHERE r.weight > r.flow)

// 2. Compute the minimum "residual capacity" along that path
WITH
  path,
  REDUCE(
    minCap = toFloat(INFINITY),
    r IN relationships(path) |
      CASE
        WHEN (r.weight - r.flow) < minCap
          THEN (r.weight - r.flow)
        ELSE
          minCap
      END
  ) AS residualCapacity

// 3. Augment: for each edge in the path, add the residualCapacity to its flow
WITH path, residualCapacity
UNWIND relationships(path) AS r
SET r.flow = r.flow + residualCapacity; "
" MATCH (source:Node_flow_8 {id: 2})-[r:CONNECTS_flow_8]->() RETURN SUM(r.flow) AS maxFlow "
```

图 20: ■

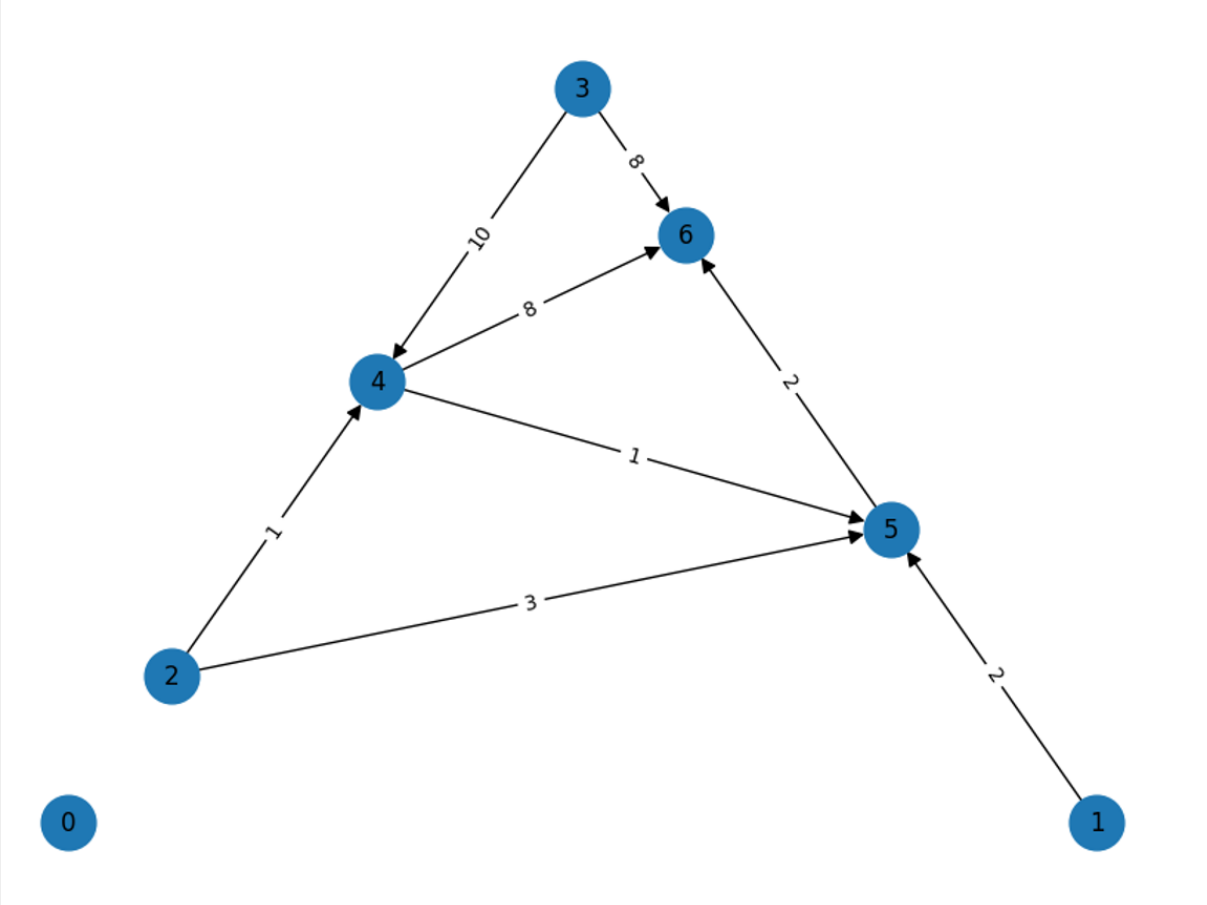

图 21: ■ 20 返回 4 作为答案。

Topological Sort

```
// 1. Find all "root" nodes (nodes with no incoming CONNECTS_topology_0 edges)
MATCH (n:Node_topology_0)
WHERE NOT (n)<-[:CONNECTS_topology_0]-()

// 2. For each root, traverse a spanning tree following outgoing CONNECTS_topology_0 relationships
CALL apoc.path.spanningTree(
  n,
  {
    relationshipFilter: 'CONNECTS_topology_0>',
    labelFilter: 'Node_topology_0',
    bfs: false
  }
) YIELD path

// 3. Extract all nodes from each tree path
WITH nodes(path) AS nds
UNWIND nds AS n

// 4. Deduplicate and collect the node IDs into a list (your "topoSort")
WITH DISTINCT n
RETURN collect(n.id) AS topoSort;
```

图 22: [redacted] Cypher [redacted]