Graph4LLM: A Systematic Survey of Graph-Enhanced Large Language Models

Xinyan Zhu[1] , **Cheng Yang**[1] , **Qiuyu Wang**[1] , **Zeyuan Guo**[1] , **Yiding Wang**[1] , **Zedi Liu**[1] , **Chunchen Wang**[1] and **Chuan Shi**[1*]

[1]Beijing University of Posts and Telecommunications

{zhuxinyan, yangcheng, autumn, guozeyuan, wangyiding, liuzedi, wangchunchen, shichuan}@bupt.edu.cn

Abstract

Large language models (LLMs) excel in natural language processing (NLP) tasks. However, they suffer from inherent limitations due to their sequence-based nature, such as structural information loss and factual unreliability. Graphs, with the ability to explicitly model entities and relations, offer an effective way to address these shortcomings. To systematically synthesize the emerging research on graph-enhanced LLMs, this survey, **Graph4LLM**, examines how these methods integrate graphs into various stages of the LLM pipeline, including the input, model, and output phases. For each phase, we provide a detailed review of the key methods and techniques. We also introduce a wide range of application scenarios where Graph4LLM methods demonstrate significant potential. Finally, we outline the challenges and future research directions for developing more efficient and interpretable solutions.

1 Introduction

Large language models (LLMs) are foundation models with billions of parameters, typically built on the Transformer architecture [Vaswani *et al.*, 2017] and pretrained on massive corpora. Under this paradigm, LLMs show impressive capabilities in natural language understanding, generation, and reasoning.

In practice, LLMs operate within a pipeline that structures information flow from input to output, as shown in Figure 1. (1) In the input phase, task specifications and external knowledge are introduced. This is done using techniques like few-shot prompting, retrieval-augmented generation (RAG) [Peng *et al.*, 2025], or by feeding the model curated training data. These methods help shape how the raw information is presented to the LLMs. (2) Next, the model focuses on processing these inputs. The Transformer-based architectures use attention mechanisms and feedforward layers to sequentially process the information, and can be further extended by multi-agent systems to coordinate multiple models through structured interactions. (3) Finally, in the output phase, LLMs generate task-specific responses. These responses can include question answering, executable code, or decision-support artifacts, which serve as the interface between model predictions and downstream applications [Zhao *et al.*, 2023].

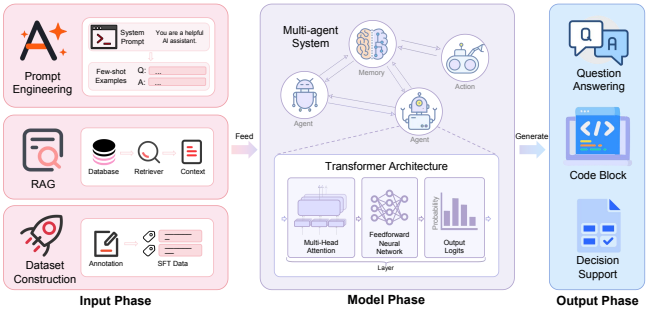

Figure 1: The overall pipeline of LLM (input phase, model phase, and output phase). The goal of our survey is to introduce: How can graphs participate in the organization of LLM inputs? How can graphs adapt the LLM architecture and collaborate multiple models? How can graphs guide the optimization of LLM outputs?

Despite their impressive performance, LLMs have inherent limitations, primarily due to their reliance on linear token sequences. Such sequential models struggle to capture complex relational structures, long-range dependencies, and multi-hop interactions, which are crucial for many knowledge-intensive tasks [Hong *et al.*, 2022]. Moreover, reasoning and planning processes are often encoded implicitly in latent representations. This makes intermediate states difficult to interpret, control, or verify systematically [Yao *et al.*, 2023]. LLMs are also vulnerable to factual inconsistencies and hallucinations, especially when tasks require precise relational reasoning or reliable knowledge grounding. These challenges highlight the inadequacy of sequence-centric models for tasks that require explicit structure, transparency, and robustness [Guan *et al.*, 2024].

To fix the problems of LLM modeling, graphs (non-Euclidean structures with nodes and edges to capture complex dependencies) and graph neural networks (GNNs, which use message-passing to learn local and global representations) [Scarselli *et al.*, 2008] provide complementary solutions. They achieve this by explicitly encoding relationships and dependencies to enable multi-hop reasoning and capture

*Corresponding author.

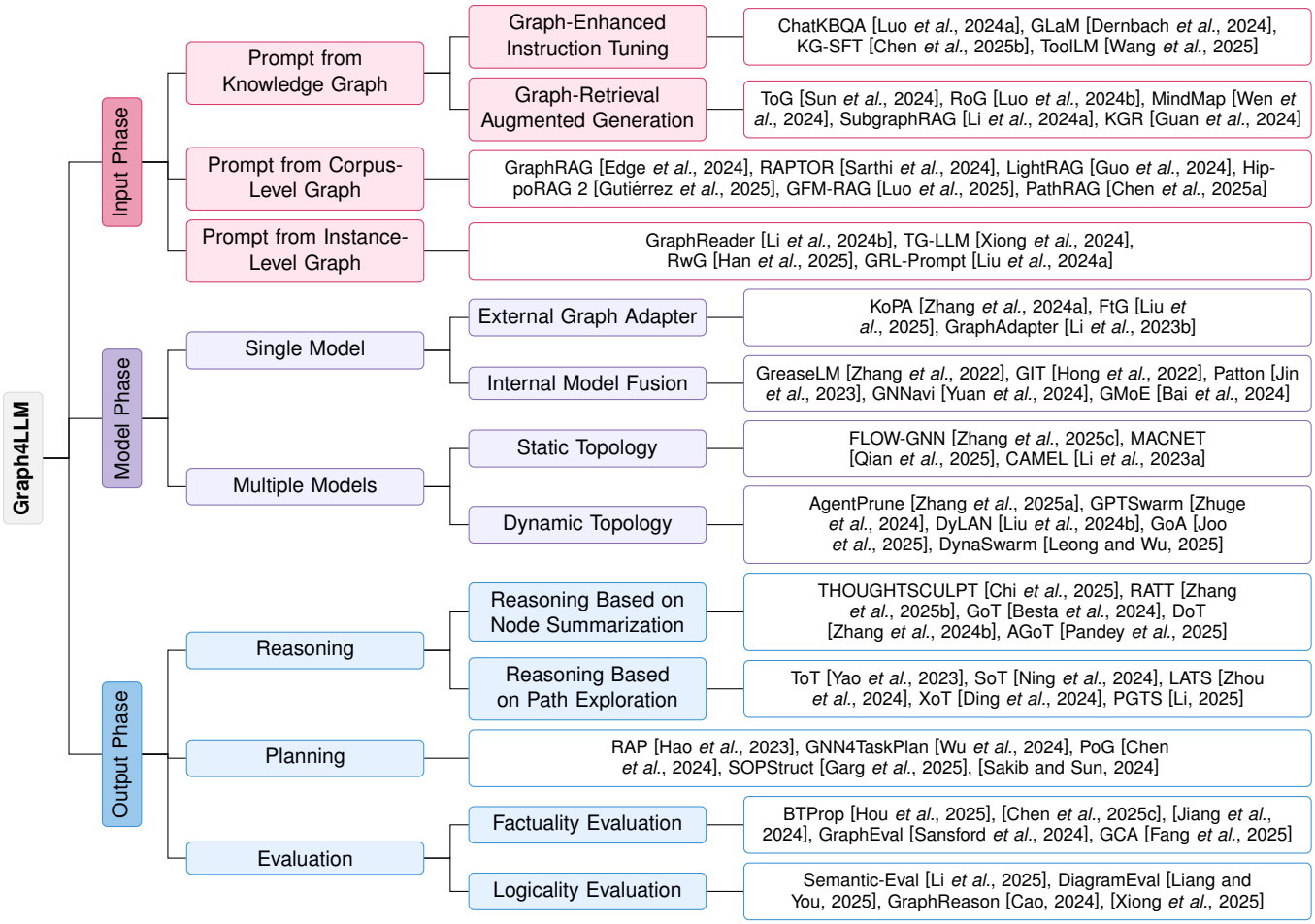

Figure 2: A taxonomy of **Graph4LLM** with representative examples.

non-linear structures. Additionally, graphs offer transparent, structured intermediate states that improve interpretability and verification. By integrating external knowledge graphs (KGs), they also enhance factual grounding, reducing hallucinations and enhancing comprehensive reliability.

Building on the potential of graphs to address LLMs limitations, graph-enhanced LLMs, which we refer to as **Graph4LLM**, leverage relational structures to handle complex, interconnected data more effectively. As a result, a growing body of work has emerged exploring different Graph4LLM methods. Despite this rapid development, research on Graph4LLM remains fragmented across communities and applications. Existing surveys often focus on narrow subtopics [Peng *et al.*, 2025], such as graph-based RAG or multi-agent systems, lacking a unified view of how graphs interact with LLMs throughout the entire pipeline.

In contrast, this paper provides the first systematic, pipeline-oriented survey of **Graph4LLM**. Specifically, we categorize existing works according to the three phases of the LLM pipeline (Figure 2): (1) In the *input phase*, graphs transform complex and scattered information into structured prompts, so that key entities and relations are clearly presented to the LLMs. (2) In the *model phase*, graphs shape the internal processing of model within a single model or organize interactions across multiple models, enabling controlled information flow and task coordination. (3) In the *output phase*, graphs reorganize LLM responses into structured representations, making intermediate steps and dependencies easy to inspect and verify. Based on this taxonomy, we further categorize each phase and present the specific methods, along with key design choices and trade-offs.

The remainder of this paper is organized as follows. Section 2 reviews multi-granularity prompt construction and knowledge incorporation techniques in the input phase. Section 3 examines single- and multi-model graph-enhanced systems in the model phase. Section 4 focuses on graph-structured reasoning, planning, and evaluation techniques in the output phase. Section 5 surveys representative Graph4LLM applications, and Section 6 discusses open challenges and future research directions. Overall, this organization follows the LLM pipeline and enables a systematic review of Graph4LLM methods.

2 Input Phase

The input phase of LLMs involves processing raw text, which is typically fed in a sequential manner. Graph4LLM input-

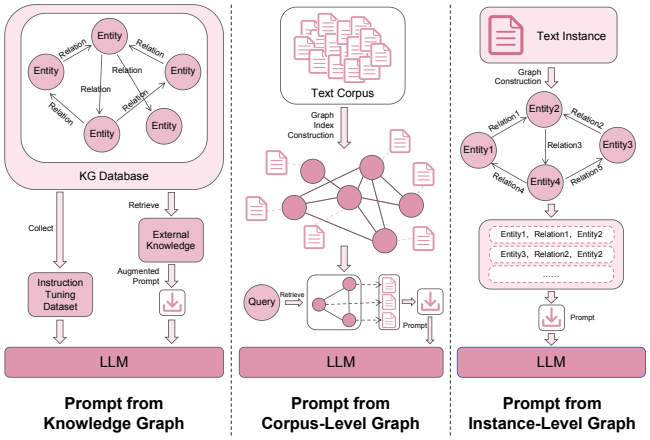

Figure 3: Different frameworks of Graph4LLM in the input phase.

phase methods extract knowledge from graphs or use them to index and organize text content. In this section, we categorize these methods based on the source of the graph structure (as shown in Figure 3): **Prompt from Knowledge Graph**, which utilizes pre-existing KGs; **Prompt from Corpus-Level Graph**, which constructs a global graph index from the text corpus; and **Prompt from Instance-Level Graph**, which induces ad-hoc structures based on specific input instances to guide the process.

2.1 Prompt from Knowledge Graph

Prompt from Knowledge Graph methods integrate existing KGs into LLM prompting pipelines to provide structured and reliable knowledge support. These methods rely on task-agnostic KGs and typically fall into two categories: Graph-Enhanced Instruction Tuning, which collects training data from KG facts and relations, and Graph-Retrieval Augmented Generation (Graph-RAG), which treats KGs as external knowledge databases for structured retrieval.

Graph-Enhanced Instruction Tuning modifies LLM parameters to align their representations with KG structures. It achieves this by constructing structure-aware instruction tuning data. This data injects explicit relational and logical information into the supervision signal, thereby enhancing the model's understanding of graphs. Existing methods mainly differ in how training data are derived from KGs. ChatKBQA [Luo *et al.*, 2024a] fine-tunes LLMs by translating SPARQL queries associated with natural language questions into logical forms. GLaM [Dernbach *et al.*, 2024] generates graph-grounded QA data by encoding node-centered k-hop neighborhoods from domain-specific KGs. KG-SFT [Chen *et al.*, 2025b] extracts reasoning subgraphs to produce question–answer explanations while addressing knowledge conflicts. ToolLM [Wang *et al.*, 2025] extends this paradigm by incorporating tool-use instruction tuning. It converts KG relations into executable APIs and generates natural language queries with corresponding solution paths.

Building on the same goal of leveraging KG structure, **Graph-Retrieval Augmented Generation (Graph-RAG)** differs by keeping LLM parameters unchanged. It retrieves task-relevant subgraphs, explicitly exploiting graph topology to provide knowledge support and enable evidence-based reasoning in real time. Existing methods mainly vary in their retrieval and reasoning strategies. ToG [Sun *et al.*, 2024] tightly couples LLMs with KGs by allowing the model to iteratively explore top-ranked reasoning paths through beam search over KG triples. RoG [Luo *et al.*, 2024b] frames reasoning as KG-grounded relation path planning, retrieving valid paths to support faithful and interpretable inference while distilling KG knowledge through targeted optimization objectives. MindMap [Wen *et al.*, 2024] adopts a prompting-based pipeline that extracts entities from inputs and constructs reasoning graphs by aggregating path-based and neighborhood evidence from KGs. To address retrieval efficiency and noise, SubgraphRAG [Li *et al.*, 2024a] introduces structure-aware subgraph retrieval with lightweight encoders and fixed in-context prompts, balancing computational cost and reasoning depth. Beyond retrieval, KGR [Guan *et al.*, 2024] incorporates iterative feedback mechanisms to verify and revise LLM-generated responses, mitigating hallucinations through KG-based self-correction.

2.2 Prompt from Corpus-Level Graph

In contrast to Prompt from Knowledge Graph, Prompt from Corpus-Level Graph constructs corpus-specific graph indices over unstructured text collections rather than relying on existing KGs. In this setting, nodes correspond to documents, entities, or concepts extracted from the corpus, and edges encode semantic or structural relations. Such graph indices enable LLMs to efficiently locate the associated textual content.

To address the lack of macro-level understanding in traditional RAG, a line of work explores global information aggregation through hierarchical structures. GraphRAG [Edge *et al.*, 2024] targets query-specific summarization over private corpora by constructing document graphs from text chunks. These graphs are organized into communities, and hierarchical summaries are generated using a Map-Reduce strategy. As a result, LLMs can integrate global context, leading to more comprehensive and diverse responses. RAPTOR [Sarthi *et al.*, 2024], on the other hand, introduces a recursive tree-based retrieval framework. By iteratively clustering text chunks, it builds a hierarchical summary tree and generates abstract summaries. This enables LLMs to retrieve high-level themes first and then access finer-grained details. Despite these advances, hierarchical aggregation can still lead to retrieval redundancy, longer reasoning chains, and limited control.

Recent work has shifted the focus from organizing information to optimizing the process of retrieval. They seek to progressively exploit and reuse these structures, improve generation efficiency and reducing retrieval noise. LightRAG [Guo *et al.*, 2024] uses graph-enhanced text indexing and dual-level retrieval to optimize reasoning over knowledge structures. It captures entity dependencies through structured graphs, while supporting incremental updates to avoid full-graph reconstruction costs. HippoRAG 2 [Gutiérrez *et al.*, 2025], inspired by the hippocampal memory system, combines LLM-based filtering with Personalized PageRank over open KGs. This graph unifies passages and phrases, enabling continual learning and associative retrieval at scale. To

address the generalization limitations of graph-enhanced retrieval, GFM-RAG [Luo *et al.*, 2025] trains a general graph foundation model (GFM) [Liu *et al.*, 2023] retriever through a two-stage process, which allows zero-shot multi-hop reasoning on unseen datasets. PathRAG [Chen *et al.*, 2025a] targets token inefficiency by pruning redundant information. It uses a flow-based algorithm to extract critical relational paths and provide ordered prompts, which improves logical coherence and reduces computational overhead.

2.3 Prompt from Instance-Level Graph

Unlike the previous two methods, Prompt from Instance-Level Graph does not use existing KGs or corpus-based index graphs. Instead, it emphasizes task-driven, on-the-fly graph construction and on-demand generation, converting a single, logically complex input instance into a graph representation. The resulting graph is then linearized into semi-structured text, which preserves the original structure and is directly fed to the LLMs.

In practical applications, researchers develop diverse graph construction and utilization strategies to address specific reasoning bottlenecks. To mitigate the "lost-in-the-middle" issue and high computational cost in long-context reasoning, GraphReader [Li *et al.*, 2024b] introduces a query-guided graph-based agent that organizes long documents into atomic fact graphs, enabling efficient multi-hop exploration under limited context windows. Focusing on temporal reasoning, TG-LLM [Xiong *et al.*, 2024] converts text into temporal graphs for graph-based inference. This process is enhanced by chain-of-thought bootstrapping and graph data augmentation, helping capture event order, duration, and inter-event relations more effectively. To address missing implicit conditions in logical reasoning, RwG [Han *et al.*, 2025] iteratively constructs and verifies explicit reasoning graphs from context and leverages them to improve multi-hop question answering. From a prompting perspective, GRL-Prompt [Liu *et al.*, 2024a] models queries and candidate demonstrations as heterogeneous graphs. It then applies reinforcement learning to explore high-order correlations, allowing for the automatic selection and arrangement of in-context examples.

3 Model Phase

The model phase of LLMs concerns both the internal architecture of the model and the way multiple agents collaborate. Graph4LLM model-phase methods introduce graph structures as explicit relational priors that complement the sequence-centric inductive bias of LLMs. Existing methods can be broadly categorized into two paradigms (as shown in Figure 4): **Single Model**, where graph modules are integrated into one LLM backbone with varying depth of fusion, and **Multiple Models**, where graphs specify or learn the communication topology and task dependencies among multiple models/agents.

3.1 Single Model

In the field of graph-enhanced single model, the core goal is to integrate structured graph signals within the LLM framework. This aims to improve the ability of model to process

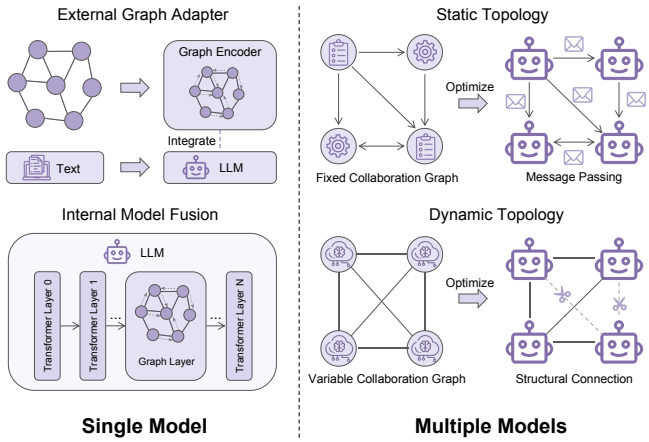

Figure 4: Different frameworks of Graph4LLM in the model phase.

relational and structural information within a unified backbone. Broadly, strategies diverge into two paradigms: External Graph Adapter, which adds graph adapters before the LLMs to fuse graph features, and Internal Model Fusion, which embeds structural interactions deeply into the model's layers for bidirectional influence.

External Graph Adapter keeps the LLM backbone largely intact while integrating graph signals through auxiliary modules. Graph structures are encoded by dedicated graph encoders, often a GNN, and mapped into the LLM representation space using lightweight adaptation mechanisms. This allows for structure-aware reasoning without altering core model parameters. Representative methods differ in how graph information is encoded and fused. KoPA [Zhang *et al.*, 2024a] incorporates pre-trained structural embeddings of KGs via a prefix-style adapter, projecting them into the model's latent space to enhance structural reasoning. FtG [Liu *et al.*, 2025] follows a filter-then-generate framework, leveraging serialized ego-graphs and a structure-text adapter to integrate graph topology while narrowing candidate entities. GraphAdapter [Li *et al.*, 2023b] extends this idea by constructing dual KGs with textual and visual substructures. Graph convolution is then applied to fuse structural knowledge into the adaptation module.

Unlike External Graph Adapter, **Internal Model Fusion** directly integrates graph layers into the internal computation of LLMs. This enables structural information to influence hidden-state updates, achieving deeper alignment between graphs and model representations. Existing methods are primarily distinguished by the integration locus of graph structure within the LLMs. A common design interleaves Transformer layers with GNN-style message passing or introduces cross-stream modules for bidirectional exchange between token and graph layers. A representative example is GreaseLM [Zhang *et al.*, 2022]. It jointly processes text and KGs using an LM and a GNN, and realizes structured interaction via dedicated interaction tokens, interaction nodes, and modality-specific interaction modules. Another integration pattern embeds graph structure directly into attention by encoding adjacency as masks, constraining information flow along graph edges or learned neighborhoods. GIT [Hong *et*

al., 2022] exemplifies this integration, with similar graph-aware Transformers proving effective when graph topology is reliable and sparsity is desirable. Beyond layer interleaving and attention control, structural priors can also be incorporated through pretraining objectives and internal routing. Patton [Jin *et al.*, 2023] introduces structure-aware pretraining on text-rich networks by combining masked language modeling with network-context objectives. GNNavi [Yuan *et al.*, 2024] inserts a GNN layer into a frozen LLM decoder, using prompt-induced graphs to guide message passing for parameter-efficient few-shot learning. GMoE [Bai *et al.*, 2024] extends this idea with a graph-routed Mixture-of-Experts architecture. It coordinates expert collaboration via GNN-based routing and mitigates load imbalance using distribution strategies.

3.2 Multiple Models

Unlike single-model methods, graph-enhanced multiple models leverage graphs to coordinate interactions, communication flows, and task dependencies across multiple models or agents. These methods are categorized into Static and Dynamic Topology. Static Topology relies on predefined fixed graph structures to ensure controllable orchestration and reduce redundant messaging. Dynamic Topology, by contrast, adapts the graph at runtime through learning, pruning, or generation mechanisms, providing greater flexibility and responsiveness in multi-agent collaboration.

Static Topology uses a fixed collaboration graph to define communication links and artifact flows in multi-agent systems. The graph is predefined based on human priors or task logic (e.g., directed acyclic pipelines, hierarchical structures, or fixed role graphs), focusing on optimizing message passing and role execution. Within this paradigm, existing methods mainly differ in the source of the predefined collaboration graph. FLOW-GNN [Zhang *et al.*, 2025c] models agentic workflows as directed acyclic graphs (DAGs), where nodes represent system instructions and edges encode task dependencies. This allows efficient performance prediction through message passing without repeated LLM invocations. Along similar lines, MACNET [Qian *et al.*, 2025] organizes large-scale multi-agent systems into DAG-structured workflows with a consistent topological order. It supports structured reasoning and scalable collaboration among over a thousand agents. Fixed topologies may also stem from system-level role designs rather than explicit task decomposition. CAMEL [Li *et al.*, 2023a] adopts a fixed role-based interaction pattern, where predefined AI assistant and AI user roles coordinate autonomous cooperation through inception prompting.

Dynamic Topology uses a variable collaboration graph that can be learned or generated at runtime. Unlike static graphs, it allows agents to dynamically adjust connections based on current inputs, states, or feedback, offering greater flexibility in multi-agent coordination. Existing methods mainly differ in how and when the collaboration structure is adjusted or constructed. AgentPrune [Zhang *et al.*, 2025a] focuses on topology sparsification by learning to remove redundant connections from dense interaction graphs. It demonstrates that sparse communication structures can outperform dense ones at a lower cost. GPTSwarm [Zhuge *et al.*, 2024] treats agents and their connections as jointly optimizable components by formulating multi-agent coordination as graph optimization over node operations and edge dataflow, while DyLAN [Liu *et al.*, 2024b] dynamically selects contributing agents and evolves communication graphs for task-oriented collaboration. Inference-time structure generation further enables query-specific collaboration graphs, where GoA [Joo *et al.*, 2025] expands agent-level interaction graphs for nonlinear context extension. And DynaSwarm [Leong and Wu, 2025] learns to select suitable communication graphs per query via reinforcement learning.

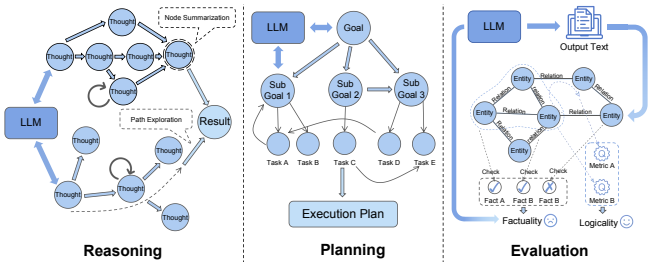

Figure 5: Different frameworks of Graph4LLM in the output phase.

4 Output Phase

The output phase of LLMs generates text sequences based on the internal representation and the received context. Graph4LLM output-phase methods represent LLM responses as graphs to clarify the thinking process, enabling more structured and verifiable outputs. We categorize existing methods into three directions based on the function of the graphs (as shown in Figure 5): **Reasoning**, which represents and traverses intermediate inference states; **Planning**, which organizes multi-step decision making and action sequences; and **Evaluation**, where graphs serve as structured references for verifying and scoring generated content.

4.1 Reasoning

Graph-based Reasoning uses graph structures to organize reasoning thoughts as relationships and dependencies between entities, improving clarity and interpretability of model outputs. Methods in this category fall into two main types: Reasoning Based on Node Summarization, which consolidates information around key nodes to guide reasoning, and Reasoning Based on Path Exploration, which conducts reasoning along sequential or branching paths.

Reasoning Based on Node Summarization organizes reasoning around central nodes that aggregate information from multiple steps. Each node acts as an information hub, condensing partial conclusions. The reasoning process converges to a final node where relevant information is summarized and synthesized into the output. Existing methods differ mainly in how reasoning nodes are structured and updated. THOUGHTSCULPT [Chi *et al.*, 2025] adopts a search-driven formulation within a Monte Carlo Tree Search (MCTS) framework. It integrates a thought generator, evaluator, and a decision simulator, while iteratively revising node-

level summaries. RATT [Zhang *et al.*, 2025b] similarly leverages tree-structured reasoning, but emphasizes path evaluation and branch selection, using node representations to identify promising reasoning trajectories. Beyond tree structures, node summarization can be extended to more general graph organizations. GoT [Besta *et al.*, 2024] frames it as a way for modeling dependencies among thoughts. It organizes reasoning steps into a graph, where edges encode inter-thought relations to support complex reasoning. DoT [Zhang *et al.*, 2024b] further operationalizes this idea with a DAG-based structure that enables node-level critique and revision of logical dependencies. AGoT [Pandey *et al.*, 2025] takes a task-decomposition perspective, breaking problems into structured sub-tasks represented as nodes. Node information then coordinates intermediate results across the decomposition hierarchy.

By comparison, **Reasoning Based on Path Exploration** frames reasoning as a traversal over multiple candidate paths. Information is not centralized at a single node but distributed along a path. Each node on the path corresponds to an independent action or intermediate decision. The final output is constructed by the accumulation of information along the selected path. Existing methods mainly differ in how paths are generated and selected during exploration. ToT [Yao *et al.*, 2023] represents reasoning as a tree and performs sequential branch exploration, explicitly comparing alternative paths to identify the most promising trajectory. Building on this formulation, SoT [Ning *et al.*, 2024] adopts a skeleton-first strategy, generating concise outlines that are later expanded in parallel to improve exploration efficiency. To further enhance path selection under long-horizon decision making, LATS [Zhou *et al.*, 2024] incorporates MCTS, tightly coupling reasoning, acting, and planning during exploration. Along a similar line, XoT [Ding *et al.*, 2024] augments MCTS with pretraining signals and external domain knowledge, enabling more flexible and informed traversal of reasoning paths. And PGTS [Li, 2025] moves toward learning-based control, integrating reinforcement learning with structured tree search to dynamically navigate reasoning paths without relying on hand-crafted heuristics.

4.2 Planning

Building on reasoning capabilities, Graph-based Planning focuses on task organization rather than inference. It leverages graphs to organize and manage interdependent goals, tasks and actions, enabling LLMs to handle complex multi-step tasks through plan execution.

Existing methods mainly differ in how they use graphs for task decomposition and execution. Typically, RAP [Hao *et al.*, 2023] treats graphs as task decomposition tools, breaking user requests into solvable sub-tasks represented as nodes with dependency edges. GNNs are then used to support sub-task retrieval and execution. Along the same line, GNN4TaskPlan [Wu *et al.*, 2024] examines how learned graph representations impact planning quality. It uses empirical analysis to explore the role of graph-based learning in optimizing decision-making for LLM-based agents. Moving beyond static task representations, PoG [Chen *et al.*, 2024] uses graphs as feedback structures during planning. These graphs incrementally construct and revise multi-level sub-goal graphs to detect deviations and enable correction through backtracking or path expansion. Furthermore, SOP-Struct [Garg *et al.*, 2025] introduces a framework where LLMs transform unstructured natural language Standard Operating Procedures (SOPs) into a structured DAG. This DAG captures logical and temporal dependencies, utilizing deterministic verification via PDDL and non-deterministic evaluation by LLMs to ensure the quality of the representation. In robotic task planning, [Sakib and Sun, 2024] utilizes graphs as execution-oriented interfaces. Multiple GPT-4–generated task trees are consolidated into a unified plan, which is then converted into executable low-level actions through cost-aware selection and symbolic planning.

4.3 Evaluation

Unlike reasoning and planning, Graph-based Evaluation focuses on assessing LLM outputs. It uses graph structures to capture richer associations among entities, facts, and sentences. Specifically, evaluation methods can be categorized into two main types: Factuality Evaluation, which assesses the accuracy and reliability of generated content through explicit entity and relation modeling; Logicality Evaluation, which measures logical coherence and reasoning consistency using graph-based relational representations.

Factuality Evaluation checks if LLM outputs meet correctness requirements by modeling content as structured graphs to verify dependencies, propagate uncertainty, and detect inconsistencies. Existing methods mainly differ in the form of evidence they construct and how it is aligned with model outputs. BTProp [Hou *et al.*, 2025] focuses on intra-output uncertainty propagation, introducing a belief-tree Markov framework that models logical relations between claims and propagates uncertainty through structured dependencies. [Chen *et al.*, 2025c] similarly operates at the single-output level but emphasizes cross-sentence contradictions and entity-level interactions to estimate factual reliability. [Jiang *et al.*, 2024] further explores uncertainty estimation by organizing claims and evidence into bipartite structures, showing that graph centrality provides more robust signals than frequency-based heuristics. Moving toward external grounding, GraphEval [Sansford *et al.*, 2024] converts generated text into structured knowledge aligned with KG triples. NLI-based verification is then incorporated to enable interpretable factual assessment. Beyond single outputs, GCA [Fang *et al.*, 2025] evaluates factuality across multiple responses. It constructs triple graphs over samples and applies RGCN-based reconstruction for black-box verification.

At a higher level, **Logicality Evaluation** assesses the quality of reasoning in LLM outputs. It goes beyond factual correctness to examine semantic coherence, inferential structure, and the organization of logical transitions using explicit relational representations. Existing methods mainly differ in which stage and granularity of structure they target. Semantic-Eval [Li *et al.*, 2025] proposes a training-free framework that constructs sentence-level semantic graphs. It applies SemanticRank to weight logical contributions and integrates pre-trained NLI models to mitigate semantic-matching bias. DiagramEval [Liang and You, 2025] targets

chart reasoning by abstracting diagrams into directed "element–relation" graphs and evaluating node and path-level alignment. For reasoning-process verification, GraphReason [Cao, 2024] merges shared intermediate steps across multiple reasoning paths into a unified graph and applies a GIN-based verifier. Beyond verification, graph-based analyses are also used to study reasoning behavior: [Xiong *et al.*, 2025] clusters semantically coherent CoT steps to construct reasoning graphs and analyze structural properties.

5 Application

Graph4LLM finds wide applications across various domains, significantly enhancing both task-level and domain-specific performance.

In **classic natural language processing (NLP)** tasks, graph structures are used to capture dependencies at different levels of language. They support a range of applications, including sequence tagging, information extraction, and text generation. By explicitly modeling syntactic, semantic, and discourse-level relationships, these graphs improve the ability to understand and generate natural language. In the realm of **code**-related applications, Graph4LLM methods operate at multiple granularities. At the function level, control-flow, data-flow, and call graphs are employed to deepen code understanding and aid in generation. On a larger scale, repository-level graphs are built to represent dependencies across files and modules, enabling cross-function reasoning, program analysis, and software maintenance tasks. For **tabular** reasoning, graphs are used to capture relationships between rows, columns, and cells. This structure facilitates multi-hop reasoning and complex query answering, going beyond the limitations of linear table formats. Graph4LLM methods are also widely utilized in **recommendation** systems. User-item interaction graphs enable LLMs to incorporate relational signals, recognizing collaborative patterns that enhance decision-making processes.

Beyond task types, Graph4LLM shows significant effectiveness in **domain**-specific applications. In medicine, graphs represent clinical knowledge and biomedical entities, helping with diagnosis and decision support. In education, concept graphs and prerequisite structures are used for curriculum planning and personalized tutoring. Financial applications benefit from relational graphs between entities, transactions, and events, which help in risk analysis and decision-making under uncertainty. In the legal field, citation and statute graphs provide a structured framework for legal reasoning and case analysis. Lastly, in electronic design automation (EDA), graphs of hardware description languages and module dependencies support LLM-based reasoning over hardware logic.

6 Challenges and Future Directions

Graph construction quality and robustness. The effectiveness of Graph4LLM methods critically depends on the quality and robustness of the constructed graphs. In practice, graphs are often derived from noisy corpora or imperfect knowledge bases, making them prone to incompleteness, spurious relations, and structural bias. These imperfections can propagate through downstream components, leading to misleading retrieval results and distorted reasoning trajectories. Future work should therefore prioritize robust and adaptive graph construction, capable of handling varying data quality and uncertainties. This can be achieved, for instance, by incorporating uncertainty quantification methods applied to graph nodes and edges, enabling LLMs to reason over probabilistic or graded relations. Additionally, iterative refinement mechanisms that leverage feedback from LLM outputs to revise or prune unreliable structures represent a promising direction.

Complexity challenges from explicit structural expansion. While richer graphs can represent more entities and relations, increasing graph size, density, or heterogeneity also introduces significant computational and cognitive challenges. In reasoning tasks, large or highly connected graphs can lead to a combinatorial explosion of paths, making search methods inefficient or unstable. Future research should focus on complexity-aware graph designs, such as structure pruning and hierarchical representations. These methods should align graph granularity with task needs, enabling the model to focus on the most relevant structures at each stage. Developing criteria to determine when and how much structure to expose to the LLMs will be key for scalable and reliable reasoning. This will help transition Graph4LLM from being "structure-rich" to "structure-effective".

Towards self-improving Graph4LLM: distillation, feedback, and co-evolution. Despite their effectiveness, most existing Graph4LLM methods use a unidirectional process, where graphs assist LLMs at specific stages but remain external to the model. The information in these graphs is consumed only during generation and not retained afterward, causing valuable structural signals to be repeatedly reconstructed, which limits efficiency and long-term knowledge transfer. To address these limitations, future work should focus on self-improving Graph4LLM systems. One promising method is graph-to-model distillation, where useful relational patterns from the Graph4LLM process are distilled into the model's internal representations through fine-tuning or knowledge distillation. This would allow the model to internalize structural priors and reduce reliance on explicit graphs during inference. Model-to-graph feedback can also refine graph construction, enabling representations to evolve based on utility and support more robust, reusable structures.

7 Conclusion

This paper presents a systematic survey of Graph4LLM, categorizing existing graph-enhanced LLM methods in the input, model, and output phases. It covers representative methods across different paradigms and their diverse application scenarios. Additionally, the paper summarizes key challenges facing current Graph4LLM research and outlines future research directions. By providing a coherent structural perspective on the integration of graphs and LLMs, this survey offers a concise overview of the field, facilitating a clear understanding of how graphs enhance the capabilities of LLMs.

References

[Bai *et al.*, 2024] Ting Bai, Yue Yu, Le Huang, et al. Gmoe: Empowering llms fine-tuning via moe graph collaboration. *arXiv:2412.16216*, 2024.

[Besta *et al.*, 2024] Maciej Besta, Nils Blach, Ales Kubicek, et al. Graph of thoughts: Solving elaborate problems with large language models. In *AAAI*, 2024.

[Cao, 2024] Lang Cao. Graphreason: Enhancing reasoning capabilities of large language models through a graph-based verification approach. In *Proceedings of the 2nd Workshop on Natural Language Reasoning and Structured Explanations (@ ACL 2024)*, 2024.

[Chen *et al.*, 2024] Liyi Chen, Panrong Tong, Zhongming Jin, et al. Plan-on-graph: Self-correcting adaptive planning of large language model on knowledge graphs. *NeurIPS*, 2024.

[Chen *et al.*, 2025a] Boyu Chen, Zirui Guo, Zidan Yang, et al. Pathrag: Pruning graph-based retrieval augmented generation with relational paths. *arXiv:2502.14902*, 2025.

[Chen *et al.*, 2025b] Hanzhu Chen, Xu Shen, Jie Wang, et al. Knowledge graph finetuning enhances knowledge manipulation in large language models. In *ICLR*, 2025.

[Chen *et al.*, 2025c] Kedi Chen, Qin Chen, Jie Zhou, et al. Enhancing uncertainty modeling with semantic graph for hallucination detection. In *AAAI*, 2025.

[Chi *et al.*, 2025] Yizhou Chi, Kevin Yang, and Dan Klein. Thoughtsculpt: Reasoning with intermediate revision and search. In *NAACL (Findings)*, 2025.

[Dernbach *et al.*, 2024] Stefan Dernbach, Khushbu Agarwal, Alejandro Zuniga, et al. Glam: Fine-tuning large language models for domain knowledge graph alignment via neighborhood partitioning and generative subgraph encoding. In *AAAI*, 2024.

[Ding *et al.*, 2024] Ruomeng Ding, Chaoyun Zhang, Lu Wang, et al. Everything of thoughts: Defying the law of penrose triangle for thought generation. In *ACL (Findings)*, 2024.

[Edge *et al.*, 2024] Darren Edge, Ha Trinh, Newman Cheng, et al. From local to global: A graph rag approach to query-focused summarization. *arXiv:2404.16130*, 2024.

[Fang *et al.*, 2025] Xinyue Fang, Zhen Huang, Zhiliang Tian, et al. Zero-resource hallucination detection for text generation via graph-based contextual knowledge triples modeling. In *AAAI*, 2025.

[Garg *et al.*, 2025] Deepeka Garg, Sihan Zeng, Sumitra Ganesh, et al. Generating structured plan representation of procedures with llms. *arXiv:2504.00029*, 2025.

[Guan *et al.*, 2024] Xinyan Guan, Yanjiang Liu, Hongyu Lin, et al. Mitigating large language model hallucinations via autonomous knowledge graph-based retrofitting. In *AAAI*, 2024.

[Guo *et al.*, 2024] Zirui Guo, Lianghao Xia, Yanhua Yu, et al. Lightrag: Simple and fast retrieval-augmented generation. *arXiv:2410.05779*, 2024.

[Gutiérrez *et al.*, 2025] Bernal Jiménez Gutiérrez, Yiheng Shu, Weijian Qi, et al. From rag to memory: Non-parametric continual learning for large language models. In *ICML*, 2025.

[Han *et al.*, 2025] Haoyu Han, Yaochen Xie, Hui Liu, et al. Reasoning with graphs: Structuring implicit knowledge to enhance llms reasoning. *arXiv:2501.07845*, 2025.

[Hao *et al.*, 2023] Shibo Hao, Yi Gu, Haodi Ma, et al. Reasoning with language model is planning with world model. In *EMNLP*, 2023.

[Hong *et al.*, 2022] Giwon Hong, Jeonghwan Kim, Junmo Kang, et al. Graph-induced transformers for efficient multi-hop question answering. In *EMNLP*, 2022.

[Hou *et al.*, 2025] Bairu Hou, Yang Zhang, Jacob Andreas, et al. A probabilistic framework for llm hallucination detection via belief tree propagation. In *NACCL*, 2025.

[Jiang *et al.*, 2024] Mingjian Jiang, Yangjun Ruan, Prasanna Sattigeri, et al. Graph-based uncertainty metrics for long-form language model generations. *NeurIPS*, 2024.

[Jin *et al.*, 2023] Bowen Jin, Wentao Zhang, Yu Zhang, et al. Patton: Language model pretraining on text-rich networks. In *ACL*, 2023.

[Joo *et al.*, 2025] Taejong Joo, Shu Ishida, Ivan Sosnovik, et al. Graph of agents: Principled long context modeling by emergent multi-agent collaboration. *arXiv:2509.21848*, 2025.

[Leong and Wu, 2025] Hui Yi Leong and Yuqing Wu. Dynaswarm: Dynamically graph structure selection for llm-based multi-agent system. *arXiv:2507.23261*, 2025.

[Li *et al.*, 2023a] Guohao Li, Hasan Hammoud, Hani Itani, et al. Camel: Communicative agents for" mind" exploration of large language model society. *NeurIPS*, 2023.

[Li *et al.*, 2023b] Xin Li, Dongze Lian, Zhihe Lu, et al. Graphadapter: Tuning vision-language models with dual knowledge graph. *NeurIPS*, 2023.

[Li *et al.*, 2024a] Mufei Li, Siqi Miao, and Pan Li. Simple is effective: The roles of graphs and large language models in knowledge-graph-based retrieval-augmented generation. *arXiv:2410.20724*, 2024.

[Li *et al.*, 2024b] Shilong Li, Yancheng He, Hangyu Guo, et al. Graphreader: Building graph-based agent to enhance long-context abilities of large language models. In *EMNLP (Findings)*, 2024.

[Li *et al.*, 2025] Shusheng Li, Jiale Li, Yifei Qu, et al. Semantic-eval: A semantic comprehension evaluation framework for large language models generation without training. In *ACL*, 2025.

[Li, 2025] Yang Li. Policy guided tree search for enhanced llm reasoning. In *ICML*, 2025.

[Liang and You, 2025] Chumeng Liang and Jiaxuan You. Diagrameval: Evaluating llm-generated diagrams via graphs. In *EMNLP*, 2025.

[Liu *et al.*, 2023] Jiawei Liu, Cheng Yang, Zhiyuan Lu, et al. Towards graph foundation models: A survey and beyond. *arXiv:2310.11829*, 2023.

[Liu *et al.*, 2024a] Yuze Liu, Tingjie Liu, Tiehua Zhang, et al. Grl-prompt: Towards knowledge graph based prompt optimization via reinforcement learning. *arXiv:2411.14479*, 2024.

[Liu *et al.*, 2024b] Zijun Liu, Yanzhe Zhang, Peng Li, et al. A dynamic llm-powered agent network for task-oriented agent collaboration. In *COLM*, 2024.

[Liu *et al.*, 2025] Ben Liu, Jihai Zhang, Fangquan Lin, et al. Filter-then-generate: Large language models with structure-text adapter for knowledge graph completion. In *COLING*, 2025.

[Luo *et al.*, 2024a] Haoran Luo, E Haihong, Zichen Tang, et al. Chatkbqa: A generate-then-retrieve framework for knowledge base question answering with fine-tuned large language models. In *ACL (Findings)*, 2024.

[Luo *et al.*, 2024b] Linhao Luo, Yuan-Fang Li, Gholamreza Haffari, et al. Reasoning on graphs: Faithful and interpretable large language model reasoning. In *ICLR*, 2024.

[Luo *et al.*, 2025] Linhao Luo, Zicheng Zhao, Gholamreza Haffari, et al. Gfm-rag: graph foundation model for retrieval augmented generation. *arXiv:2502.01113*, 2025.

[Ning *et al.*, 2024] Xuefei Ning, Zinan Lin, Zixuan Zhou, et al. Skeleton-of-thought: Prompting llms for efficient parallel generation. In *ICLR*, 2024.

[Pandey *et al.*, 2025] Tushar Pandey, Ara Ghukasyan, Oktay Goktas, et al. Adaptive graph of thoughts: Test-time adaptive reasoning unifying chain, tree, and graph structures. *arXiv:2502.05078*, 2025.

[Peng *et al.*, 2025] Boci Peng, Yun Zhu, Yongchao Liu, et al. Graph retrieval-augmented generation: A survey. *ACM Transactions on Information Systems*, 2025.

[Qian *et al.*, 2025] Chen Qian, Zihao Xie, YiFei Wang, et al. Scaling large language model-based multi-agent collaboration. In *ICLR*, 2025.

[Sakib and Sun, 2024] Md Sadman Sakib and Yu Sun. Consolidating trees of robotic plans generated using large language models to improve reliability. *arXiv:2401.07868*, 2024.

[Sansford *et al.*, 2024] Hannah Sansford, Nicholas Richardson, Hermina Petric Maretic, et al. Grapheval: A knowledge-graph based llm hallucination evaluation framework. *arXiv:2407.10793*, 2024.

[Sarthi *et al.*, 2024] Parth Sarthi, Salman Abdullah, Aditi Tuli, et al. Raptor: Recursive abstractive processing for tree-organized retrieval. In *ICLR*, 2024.

[Scarselli *et al.*, 2008] Franco Scarselli, Marco Gori, Ah Chung Tsoi, et al. The graph neural network model. *IEEE transactions on neural networks*, 2008.

[Sun *et al.*, 2024] Jiashuo Sun, Chengjin Xu, Lumingyuan Tang, et al. Think-on-graph: Deep and responsible reasoning of large language model on knowledge graph. In *ICLR*, 2024.

[Vaswani *et al.*, 2017] Ashish Vaswani, Noam Shazeer, Niki Parmar, et al. Attention is all you need. *NeurIPS*, 2017.

[Wang *et al.*, 2025] Jingwei Wang, Zai Zhang, Hao Qian, et al. Enhancing llm tool use with high-quality instruction data from knowledge graph. *arXiv:2506.21071*, 2025.

[Wen *et al.*, 2024] Yilin Wen, Zifeng Wang, and Jimeng Sun. Mindmap: Knowledge graph prompting sparks graph of thoughts in large language models. In *ACL*, 2024.

[Wu *et al.*, 2024] Xixi Wu, Yifei Shen, Caihua Shan, et al. Can graph learning improve planning in llm-based agents? *NeurIPS*, 2024.

[Xiong *et al.*, 2024] Siheng Xiong, Ali Payani, Ramana Kompella, et al. Large language models can learn temporal reasoning. In *ACL*, 2024.

[Xiong *et al.*, 2025] Zhen Xiong, Yujun Cai, Zhecheng Li, et al. Mapping the minds of llms: A graph-based analysis of reasoning llm. *arXiv:2505.13890*, 2025.

[Yao *et al.*, 2023] Shunyu Yao, Dian Yu, Jeffrey Zhao, et al. Tree of thoughts: Deliberate problem solving with large language models. *NeurIPS*, 2023.

[Yuan *et al.*, 2024] Shuzhou Yuan, Ercong Nie, Michael Färber, et al. Gnnavi: Navigating the information flow in large language models by graph neural network. In *ACL (Findings)*, 2024.

[Zhang *et al.*, 2022] Xikun Zhang, Antoine Bosselut, Michihiro Yasunaga, et al. Greaselm: Graph reasoning enhanced language models for question answering. In *ICLR*, 2022.

[Zhang *et al.*, 2024a] Yichi Zhang, Zhuo Chen, Lingbing Guo, et al. Making large language models perform better in knowledge graph completion. In *ACM Multimedia*, 2024.

[Zhang *et al.*, 2024b] Yifan Zhang, Yang Yuan, and Andrew Chi-Chih Yao. On the diagram of thought. *arXiv:2409.10038*, 2024.

[Zhang *et al.*, 2025a] Guibin Zhang, Yanwei Yue, Zhixun Li, et al. Cut the crap: An economical communication pipeline for llm-based multi-agent systems. In *ICLR*, 2025.

[Zhang *et al.*, 2025b] Jinghan Zhang, Xiting Wang, Weijieying Ren, et al. Ratt: A thought structure for coherent and correct llm reasoning. In *AAAI*, 2025.

[Zhang *et al.*, 2025c] Yuanshuo Zhang, Yuchen Hou, Bohan Tang, et al. Gnns as predictors of agentic workflow performances. *arXiv:2503.11301*, 2025.

[Zhao *et al.*, 2023] Wayne Xin Zhao, Kun Zhou, Junyi Li, et al. A survey of large language models. *arXiv:2303.18223*, 2023.

[Zhou *et al.*, 2024] Andy Zhou, Kai Yan, Michal Shlapentokh-Rothman, et al. Language agent tree search unifies reasoning, acting, and planning in language models. In *ICML*, 2024.

[Zhuge *et al.*, 2024] Mingchen Zhuge, Wenyi Wang, Louis Kirsch, et al. Gptswarm: Language agents as optimizable graphs. In *ICML*, 2024.