International Journal of Trendy Research in Engineering and Technology
Volume 10 Issue 3 June 2026
ISSN No. 2582-0958

LARGE LANGUAGE MODEL-ENABLED GRAPH ALGORITHMS AND GRAPH COMPUTING: TECHNIQUES AND APPLICATIONS

Zohaib Ali
Department of Computer Science, The Islamia University of Bahawalpur, Bahawalpur, Pakistan
Email: f21bdocs1m01027@iub.edu.pk, chzohaib1027ali@gmail.com

Received 25 June 2026 Received in revised form 02 July 2026 Accepted 08 July 2026
Available Online 10 July 2026

ABSTRACT

Graphs are a fundamental structure for representing relational information across domains such as social networks, molecular structures, and knowledge bases, and graph computing systems and algorithms (e.g., PageRank, shortest-path search, community detection, graph neural networks) have long been central to extracting insight from such data. The recent emergence of large language models (LLMs) has opened a new frontier in graph computing, enabling natural-language interfaces to graph databases, LLM-augmented graph representation learning, and retrieval-augmented generation over graph-structured knowledge. This paper surveys the current landscape of LLM-enabled graph algorithms and graph computing, organizing existing approaches into four paradigms: graph-to-text linearization for direct LLM reasoning, LLM-augmented graph neural networks, graph-based retrieval-augmented generation (GraphRAG), and LLM agents that orchestrate external graph-processing tools. We illustrate each paradigm with schematic architecture diagrams, review applications across knowledge graph question answering, social network analysis, scientific discovery, software engineering, and urban analytics, and discuss open challenges including scalability under limited context windows, hallucination and faithfulness of LLM-generated graph reasoning, and the current state of algorithmic graph-reasoning benchmarks. We conclude with promising future directions, including graph foundation models and tighter integration between LLM reasoning and classical graph algorithms.

Keywords: large language models; graph computing; graph algorithms; graph neural networks; retrieval-augmented generation; knowledge graphs

1 INTRODUCTION

Graphs, which are structures of nodes and edges that encode relationships, are one of the most general representations of real-world data, spanning social networks, transportation systems, molecular structures, knowledge bases, and software dependency structures. Decades of research have produced mature graph computing infrastructure, from classical traversal and shortest-path algorithms to distributed graph-processing systems and, more recently, graph neural networks (GNNs) that learn representations directly from graph structure and node features [1], [2].

Separately, large language models (LLMs) have rapidly advanced natural language understanding, in-context reasoning, and code generation, and are now being deployed across information retrieval, recommendation, and scientific domains. However, LLMs are fundamentally sequence-based models: they do not natively perceive graph topology, and reasoning correctly about large, structurally complex graphs remains difficult for even the most capable current models. This mismatch between the relational nature of graph data and the sequential nature of LLMs has become a highly active research frontier, with a dedicated workshop series (LLM+Graph, co-located with VLDB) [3] and multiple comprehensive surveys published within the past two years [4], [5].

This paper reviews this emerging area from a practitioner's perspective. Rather than presenting new experimental results, our contribution is a structured synthesis of the field intended to orient researchers and practitioners who are approaching LLM-graph integration for the first time. Specifically, this paper:

1. Organizes existing LLM-graph integration approaches into a four-paradigm taxonomy (Section 3), clarifying when each is appropriate, and illustrates each paradigm with a schematic architecture diagram.
2. Surveys concrete algorithmic techniques for LLM-enabled graph computing, including natural-language-to-graph-query translation and LLM-assisted algorithm design (Section 4).
3. Reviews representative application domains and the specific open challenge each domain faces (Section 5), and summarizes emerging evaluation benchmarks (Section 6).
4. Discusses cross-cutting challenges, including scalability, hallucination, and evaluation, and outlines promising future directions (Sections 7-8).

2 BACKGROUNDS

2.1 Graph Computing Fundamentals

A graph G = (V, E) consists of a vertex set V and an edge set E describing pairwise relationships. Classical graph algorithms, including breadth-first and depth-first search, shortest-path algorithms (e.g., Dijkstra's), PageRank, and community detection, form the algorithmic core of graph computing and remain widely used in production systems, as detailed in standard references on large-scale data mining [6]. At scale, distributed graph-processing frameworks such as Pregel popularized the vertex-centric "think-like-a-vertex" programming model for processing graphs with billions of edges [7].

Graph neural networks (GNNs) extend this landscape by learning vector representations of nodes and edges directly from graph structure and associated features, rather than relying on hand-designed algorithms. Foundational architectures include the graph convolutional network (GCN), which generalizes convolution to graph-structured data [1], and GraphSAGE, which introduced inductive, sample-based neighborhood aggregation for large graphs [2]. These learning-based methods now underpin most modern recommendation, fraud-detection, and molecular-property-prediction systems.

2.2 Large Language Models: Capabilities and Limitations for Structured Data

Modern LLMs are built on the transformer architecture [8] and, when scaled to billions of parameters and trained on large text corpora, exhibit strong in-context learning and few-shot generalization capabilities [9]. These capabilities make LLMs attractive for tasks that were previously the domain of hand-crafted natural-language interfaces, including translating natural-language questions into structured database queries.

However, LLMs process input as linear token sequences and have no native mechanism for perceiving graph topology; a graph must be serialized into text before an LLM can reason about it, and this serialization is lossy and consumes a large fraction of the model's limited context window as graph size grows. Recent benchmarking work evaluating LLMs directly on algorithmic graph-reasoning tasks (e.g., connectivity, shortest path, cycle detection) finds that performance degrades substantially as graph size and structural complexity increase, even for the strongest current models [10]. This limitation directly motivates the hybrid paradigms discussed in Section 3.

2.3 Scope and Methodology of This Review

This review focuses specifically on the intersection of LLMs and graph-structured data processing, and does not attempt to cover LLM research or graph algorithm research in isolation except where necessary for background. Source material was identified through the dedicated LLM+Graph workshop series, recent (2024-2025) survey articles on LLMs for graphs, and widely cited foundational papers on graph neural networks and transformer-based language models. Given the pace of publication in this area, this review should be read as a snapshot of an actively evolving field rather than a final taxonomy; Section 8 discusses directions likely to reshape this landscape.

3 PARADIGMS FOR LLM-ENABLED GRAPH COMPUTING

Recent surveys converge on a small number of recurring strategies for combining LLMs with graph-structured data [4], [5]. Figure 1 summarizes the four paradigms discussed in this section as a taxonomy, and Table 1 lists their core idea and representative work.

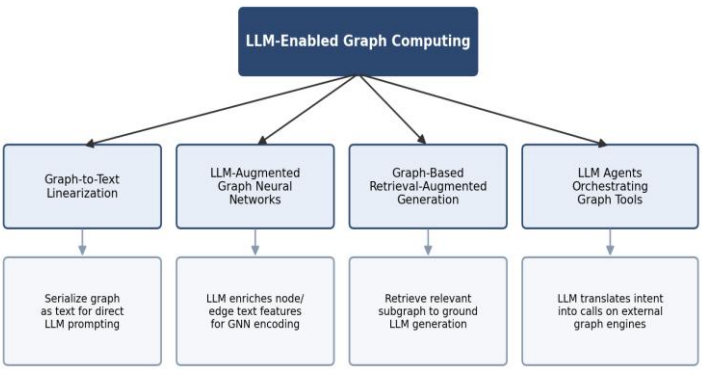

Figure 1. Taxonomy of the four paradigms for LLM-enabled graph computing discussed in this survey.

Table 1. Taxonomy of LLM-enabled graph computing paradigms.

Paradigm	Core Idea	Representative Work
Graph-to-text linearization	Serialize graph structure (nodes, edges, attributes) into text so the LLM can reason over it directly via prompting.	Xypolopoulos et al., 2025 [11]
LLM-augmented GNNs	Use an LLM to generate or enrich textual node/edge features before or alongside graph neural network encoding.	Ren et al., 2024 [4]
Graph-based retrieval-augmented generation (GraphRAG)	Ground LLM generation in a retrieved subgraph or community structure from a knowledge graph, rather than flat text chunks.	GFM-RAG, 2025 [12]; ArchRAG, 2025
LLM agents orchestrating graph tools	LLM translates natural-language requests into calls on external graph engines (e.g., text-to-Cypher/SPARQL) rather than performing graph computation itself.	VLDB'25 LLM+Graph Workshop [3]

3.1 Graph-to-Text Linearization

The most direct paradigm converts a graph into a textual representation, such as an edge list, adjacency list, or natural-language description of nodes and relationships, which is then included in an LLM prompt (Figure 2). Recent work has systematically compared linearization schemes and found that the choice of serialization format materially affects downstream reasoning accuracy, independent of model scale [11]. This paradigm requires no model training and is the easiest to deploy, but scales poorly: linearized representations grow with graph size, quickly exceeding practical context-window budgets for large graphs.

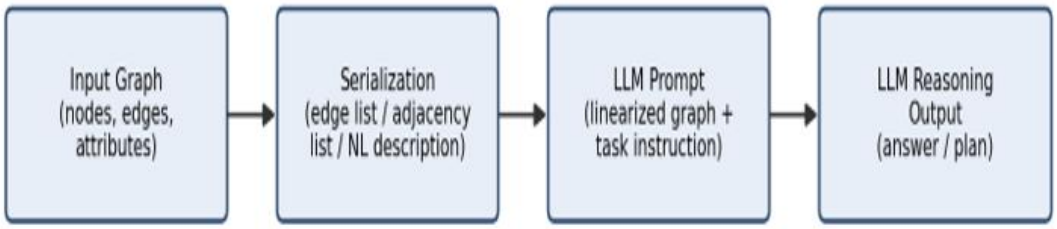

Figure 2. Graph-to-text linearization pipeline: a graph is serialized into text before being included in an LLM prompt.

3.2 LLM-Augmented Graph Neural Networks

A second paradigm keeps a conventional GNN as the graph-reasoning backbone but uses an LLM to generate or enrich textual node and edge features beforehand, for example by generating richer natural-language descriptions for nodes in a text-attributed graph, or by imputing missing textual attributes (Figure 3). This preserves the scalability of GNN-based graph computation while injecting the LLM's broader world knowledge and language understanding into the node/edge representations [4].

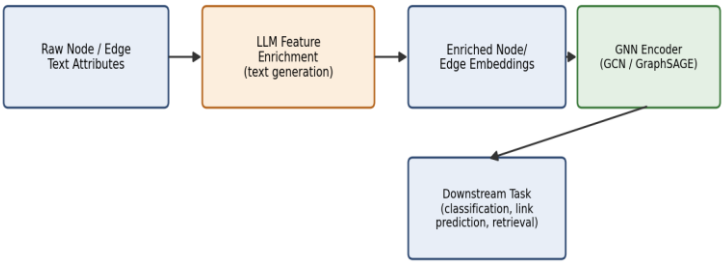

Figure 3. LLM-augmented GNN architecture: an LLM enriches node/edge text before conventional GNN encoding and downstream prediction.

3.3 Graph-Based Retrieval-Augmented Generation (GraphRAG)

Rather than retrieving flat text chunks as in conventional retrieval-augmented generation, GraphRAG approaches retrieve a relevant subgraph, entity neighborhood, or community structure from a knowledge graph and provide it as grounding context for LLM generation (Figure 4). This has been shown to improve multi-hop reasoning quality compared with flat-text retrieval, and has motivated dedicated graph foundation models for retrieval, such as GFM-RAG, and hierarchical, community-based retrieval methods such as ArchRAG [12], [13].

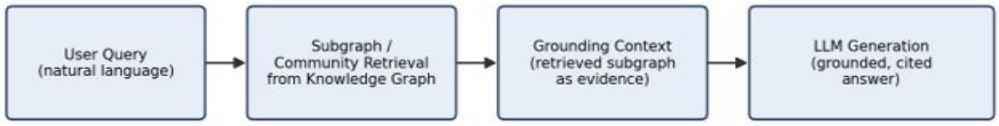

Figure 4. GraphRAG pipeline: a subgraph retrieved from a knowledge graph grounds the LLM's generated answer.

3.4 LLM Agents Orchestrating External Graph Tools

The fourth paradigm treats the LLM not as the graph-reasoning engine itself, but as a natural-language interface that translates user intent into calls on an external, purpose-built graph-processing system (e.g., a graph database or graph analytics engine), as shown in Figure 5. Practitioners at the second LLM+Graph workshop (co-located with VLDB 2025) broadly agreed that LLMs are best positioned as natural-language interfaces, for instance translating text into graph query languages, rather than as replacements for core graph database functionality [3]. This paradigm tends to be the most reliable for large-scale, production graph systems, since exact graph computation is delegated to a system designed for it.

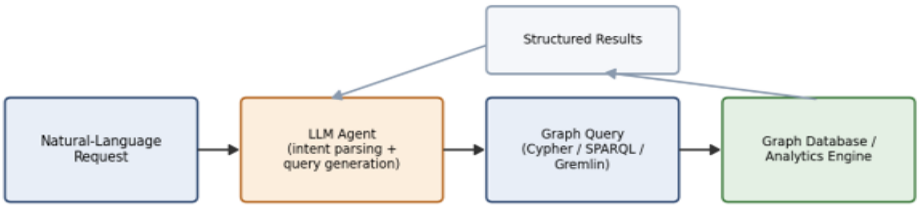

Figure 5. LLM agent orchestration: natural-language requests are translated into formal graph queries executed by an external graph engine.

3.5 Comparing the Four Paradigms

The four paradigms trade off scalability to large graphs against the flexibility of LLM reasoning they permit. Figure 6 positions them conceptually along these two qualitative axes; this positioning reflects the design intent and reported characteristics of each paradigm in the surveyed literature, and is illustrative rather than derived from a controlled empirical comparison, since no single benchmark currently evaluates all four paradigms under matched conditions.

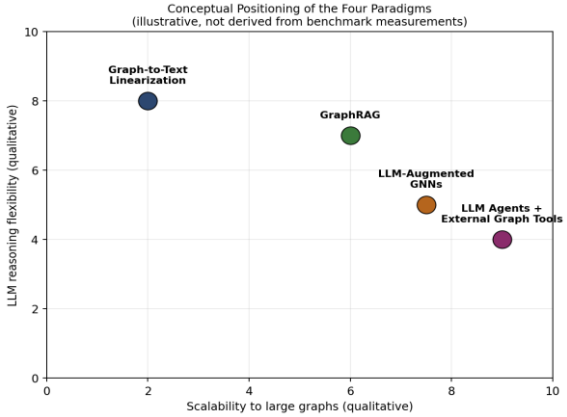

Figure 6. Conceptual positioning of the four paradigms by scalability and LLM reasoning flexibility (illustrative, not an empirical measurement; original synthesis by the author based on the surveyed literature).

In practice, graph-to-text linearization offers the most direct access to LLM reasoning but the weakest scalability; LLM agents orchestrating external graph tools offer the strongest scalability by delegating computation to purpose-built systems, at the cost of constraining the LLM to an interface role; and LLM-augmented GNNs and GraphRAG occupy intermediate positions, each retaining a conventional graph-computation backbone while incorporating LLM-derived semantic enrichment. Illustrative production-oriented systems such as GraphSeek combine elements of several of these paradigms in practice [14].

4 LLM-ENABLED GRAPH ALGORITHMS

4.1 Natural-Language-to-Graph-Query Translation

A practical and increasingly mature application is translating natural-language requests into formal graph query languages such as Cypher or SPARQL (text-to-query), lowering the barrier for non-expert users to query knowledge graphs and graph databases. This is a direct instance of the agent-orchestration paradigm (Section 3.4) and is one of the few LLM-graph integration patterns already seeing production deployment. Correctness of the generated query, rather than fluency of the natural-language exchange, is the primary evaluation criterion in this setting.

4.2 LLM-Assisted Algorithm Design and Code Generation

LLMs' code-generation capabilities can be applied to graph algorithm design itself: generating, explaining, or optimizing implementations of classical graph algorithms, or proposing candidate heuristics for problem variants without known efficient exact solutions. Dedicated benchmarks such as GTA (Graph Theory Agent) have been introduced specifically to evaluate LLMs' algorithmic graph-reasoning ability in an agentic setting, rather than as a one-shot question-answering task [10].

4.3 LLM-in-the-Loop Heuristic Search for Hard Graph Problems

For NP-hard graph problems (e.g., graph coloring, maximum clique, the traveling salesman problem), a growing line of work uses an LLM to propose or refine heuristics within an outer search or optimization loop, combining the LLM's pattern-recognition and code-generation ability with a classical solver's correctness guarantees. This hybrid design mirrors the LLM-augmented GNN paradigm in spirit: the LLM contributes knowledge and creativity, while a conventional algorithmic component ensures correctness and scalability.

4.4 Summary of Algorithmic Techniques

Across Sections 4.1-4.3, a common pattern emerges: the LLM is most effective, and most reliable, when its role is scoped to a genuinely language-dependent sub-task (interpreting intent, generating a candidate query or heuristic, explaining a result), while exact computation, verification, or execution is left to conventional, well-understood graph algorithms and systems. This scoping principle recurs throughout Sections 5-7 of this survey.

5 APPLICATIONS

Table 2 summarizes representative application domains for LLM-enabled graph computing, alongside the primary open challenge in each; each is discussed in more detail below.

Table 2. Representative application domains for LLM-enabled graph computing.

Domain	Example Use Case	Key Open Challenge
Knowledge graphs & QA	Multi-hop question answering grounded in enterprise or encyclopedic knowledge graphs.	Faithfulness / hallucination
Social network analysis	Community detection and influence analysis enriched with natural-language user/content context.	Scalability to billions of edges
Scientific discovery	Reasoning over molecular graphs and biological pathway graphs for hypothesis generation.	Domain-specific correctness
Software engineering	Reasoning over dependency and call graphs for bug localization and refactoring suggestions.	Very large, dynamic graphs
Urban & spatial analytics	Constructing and querying urban knowledge graphs (transport, points of interest) via LLM agents.	Real-time graph updates

5.1 Knowledge Graphs and Question Answering

Knowledge-graph question answering is the most mature application area, with substantial recent work on re-scoring and injecting knowledge-graph facts into LLM generation, and on faithful, well-formed multi-hop reasoning chains grounded in graph structure rather than free-form generation. GraphRAG-style grounding (Section 3.3) is currently the dominant approach in this domain, since it directly addresses the faithfulness concerns that arise when an LLM answers questions from parametric knowledge alone.

5.2 Social Network Analysis and Recommendation

LLMs can enrich social and interaction graphs with natural-language understanding of user-generated content, supporting community detection and recommendation systems that combine structural signals with semantic understanding of text. The LLM-augmented GNN paradigm (Section 3.2) is a natural fit here, since production recommendation systems typically already rely on GNN-based architectures, and scalability to graphs with billions of edges remains an open systems challenge that limits how much LLM computation can be introduced per node or edge.

5.3 Scientific Discovery

In biology and chemistry, graph-structured data, including molecular graphs and biological pathway graphs, are a natural fit for LLM-graph integration. Dedicated benchmarks such as BioMaze have been developed specifically to evaluate and improve LLM reasoning over biological pathway graphs, reflecting the domain-specific correctness demands of scientific applications, where an incorrect but plausible-sounding answer can be far more costly than in general-purpose question answering.

5.4 Software Engineering

Source-code dependency graphs and function call graphs are large, dynamic graphs that are natural candidates for LLM-assisted analysis, including bug localization and refactoring suggestions grounded in structural context rather than local code snippets alone. Because these graphs change with every commit, LLM agents that query a live, incrementally updated graph representation (Section 3.4) are generally more practical than approaches that require re-linearizing the entire graph for each query.

5.5 Urban and Spatial Analytics

Urban knowledge graphs, which link transportation networks, points of interest, and administrative regions, have motivated dedicated LLM agent frameworks for automated urban knowledge graph construction and querying, an area of active systems and applications research. Real-time updates (e.g., traffic conditions) place similar demands on this application as the dynamic software-dependency-graph setting in Section 5.4.

6 EVALUATION BENCHMARKS

Systematic evaluation of LLM graph-reasoning ability is an active and still-maturing area. Table 3 summarizes three representative benchmarking efforts discussed in this survey.

Table 3. Representative benchmarks for evaluating LLM-graph reasoning.

Benchmark	Focus	Reference
GTA (Graph Theory Agent)	Agentic, algorithmic graph reasoning (connectivity, shortest path, cycle detection) rather than one-shot QA.	Xu et al., 2025 [10]
BioMaze	LLM reasoning over biological pathway graphs for scientific hypothesis tasks.	Domain benchmark, 2025
Graph-linearization comparison suites	Systematic comparison of serialization formats (edge list, adjacency list, natural-language) and their effect on reasoning accuracy.	Xypolopoulos et al., 2025 [11]

A common theme across these benchmarks is the attempt to isolate algorithmic or structural graph reasoning from surface-level pattern matching that a large model may perform without genuinely tracking graph structure. However, the field currently lacks a single, widely agreed-upon benchmark suite comparable to those available for classical graph algorithms (e.g., standard shortest-path or PageRank test suites), which complicates direct comparison across the paradigms discussed in Section 3.

7 CHALLENGES AND OPEN PROBLEMS

7.1 Scalability and Context-Length Limits

Graph linearization (Section 3.1) fundamentally does not scale: as graphs grow, their textual serialization quickly exceeds practical LLM context budgets, and even within budget, reasoning accuracy degrades with graph size and structural complexity [10]. Hybrid paradigms that delegate exact computation to conventional graph systems (Sections 3.2-3.4) are the primary mitigation currently available.

7.2 Hallucination and Faithfulness

LLMs can generate plausible-sounding but structurally incorrect graph relationships or query results, a concern that has motivated dedicated work specifically at the intersection of knowledge graphs, LLMs, and hallucination. Grounding generation in retrieved, verifiable subgraph evidence (GraphRAG) is currently the most established mitigation strategy, though it does not eliminate the problem entirely.

7.3 Evaluation Benchmarks

As discussed in Section 6, recent benchmarks such as GTA and graph-reasoning-structured question-answering datasets aim to isolate algorithmic graph reasoning from surface-level pattern matching, but the field currently lacks a single, widely agreed-upon benchmark suite comparable to those available for classical graph algorithms.

7.4 Efficiency and Deployment Cost

LLM inference is substantially more computationally expensive than classical graph algorithms, and repeated LLM calls within an agentic loop (Sections 3.4, 4.3) compound this cost. Efficient deployment likely requires restricting LLM involvement to genuinely language-dependent sub-tasks (e.g., interpreting a query, generating a heuristic) while leaving bulk graph computation to conventional, optimized systems.

8 FUTURE DIRECTIONS

Graph foundation models, which are models pre-trained to transfer across graph domains and tasks analogous to how LLMs transfer across language tasks, have recently been proposed as a unifying direction for the field, though substantial conceptual and practical challenges remain in defining a graph analogue of the token vocabulary and pre-training objectives that made language foundation models successful [15]. Other promising directions include tighter co-design between LLM reasoning and classical graph algorithms (rather than treating them as separate stages), standardized, difficulty-graded benchmarks for algorithmic graph reasoning, and multimodal graphs that combine text, structure, and other modalities such as images or time series within a single graph representation.

9 LIMITATIONS OF THIS SURVEY

This review has three main limitations. First, it synthesizes a fast-moving literature; several cited works are recent preprints that may be revised or formally published elsewhere by the time of reading (see the note preceding the reference list). Second, the positioning in Figure 6 and the paradigm boundaries in Section 3 are the author's synthesis of the surveyed literature rather than an outcome of a systematic, pre-registered literature review protocol; a systematic mapping study would be a valuable complement to this narrative synthesis. Third, this survey does not include original experimental benchmarking of the paradigms discussed; where quantitative claims are made, they are attributed to the cited source rather than independently verified here.

10 CONCLUSION

This paper surveyed the emerging area of LLM-enabled graph algorithms and graph computing, organizing current approaches into four paradigms, namely graph-to-text linearization, LLM-augmented GNNs, graph-based retrieval-augmented generation, and LLM agents orchestrating external graph tools, illustrating each with a schematic architecture diagram, and reviewed their application across knowledge graphs, social networks, scientific discovery, software engineering, and urban analytics. While LLMs bring valuable natural-language understanding and reasoning flexibility to graph computing, current evidence indicates they are best deployed as a complement to, rather than a replacement for, conventional graph algorithms and systems, particularly at scale. Closing the gap between LLM reasoning and reliable, scalable graph computation remains an open and active research direction.

ACKNOWLEDGMENT

The author declares no conflicts of interest and received no dedicated funding for this work.

REFERENCES

[1] T. N. Kipf and M. Welling, "Semi-Supervised Classification with Graph Convolutional Networks," in Proc. Int. Conf. Learning Representations (ICLR), Toulon, France, 2017.

[2] W. L. Hamilton, R. Ying, and J. Leskovec, "Inductive Representation Learning on Large Graphs," in Proc. 31st Conf. Neural Information Processing Systems (NeurIPS), Long Beach, CA, USA, 2017.

[3] "LLM+Graph@VLDB'2025 Workshop Summary," 2nd Int. Workshop on Large Language Models and Graphs, co-located with the 51st Int. Conf. Very Large Data Bases (VLDB), London, U.K., 2025.

[4] X. Ren, J. Tang, D. Yin, N. Chawla, and C. Huang, "A Survey of Large Language Models for Graphs," in Proc. 30th ACM SIGKDD Conf. Knowledge Discovery and Data Mining (KDD), Barcelona, Spain, 2024.

[5] "A Survey of Large Language Models for Data Challenges in Graphs," Expert Systems with Applications, 2025.

[6] J. Leskovec, A. Rajaraman, and J. D. Ullman, Mining of Massive Datasets, 3rd ed. Cambridge, U.K.: Cambridge Univ. Press, 2020.

[7] G. Malewicz, M. H. Austern, A. J. C. Bik, J. C. Dehnert, I. Horn, N. Leiser, and G. Czajkowski, "Pregel: A System for Large-Scale Graph Processing," in Proc. ACM SIGMOD Int. Conf. Management of Data, Indianapolis, IN, USA, 2010, pp. 135-146.

[8] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention Is All You Need," in Proc. 31st Conf. Neural Information Processing Systems (NeurIPS), Long Beach, CA, USA, 2017.

[9] T. B. Brown et al., "Language Models Are Few-Shot Learners," in Proc. 34th Conf. Neural Information Processing Systems (NeurIPS), Vancouver, Canada, 2020.

[10] Z. Xu, Y. Wang, Y. Huang, C. Wang, L. Gao, Z. Song, Z. Chen, X. Zhang, and X. Chen, "GTA: Graph Theory Agent and Benchmark for Algorithmic Graph Reasoning with LLMs," arXiv preprint, 2025.

[11] C. Xypolopoulos, G. Shang, X. Fei, G. Nikolentzos, H. Abdine, I. Evdaimon, M. Chatzianastasis, G. Stamou, and M. Vazirgiannis, "Graph Linearization Methods for Reasoning on Graphs with Large Language Models," arXiv:2410.19494, 2025.

[12] "GFM-RAG: Graph Foundation Model for Retrieval Augmented Generation," arXiv preprint, 2025.

[13] "A Survey of Graph Retrieval-Augmented Generation for Customized Large Language Models," arXiv preprint, 2025.

[14] "GraphSeek: Next-Generation Graph Analytics with LLMs," arXiv preprint.

[15] J. Liu, C. Yang, Z. Lu, J. Chen, Y. Li, M. Zhang, T. Bai, Y. Fang, L. Sun, P. S. Yu, and C. Shi, (2025)"Graph Foundation Models: Concepts, Opportunities and Challenges," IEEE Trans. Pattern Analysis and Machine Intelligence, 47(6) 5023-5044, .